Advanced Far Field EM Side-Channel Attack on AES

Ruize Wang
ruize@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

Huanyu Wang*
huanyu@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

Elena Dubrova
dubrova@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

Martin Brisfors
brisfors@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

ABSTRACT

Several attacks on AES using far field electromagnetic (EM) emission as a side channel have been recently presented. Unlike power analysis or near filed EM analysis, far field EM attacks do not require a close physical proximity to the device under attack. However, in all previous attacks traces for the profiling stage are also captured at a distance (fixed or variable) from the profiling devices. This degenerates the quality of profiling traces due to noise and interference. In this paper, we train deep learning models on "clean" traces, captured through a coaxial cable. Our experiments show that the resulting models can extract the AES key from less than 500 traces on average captured at 15 m from the victim device without repeating each encryption more than once. This is a 20-fold improvement over the previous attacks which require about 10K traces for the same conditions.

CCS CONCEPTS

• **Security and privacy → Cryptography**; **Cryptanalysis and other attacks**;

KEYWORDS

Side-channel analysis, far field EM emissions, profiled attack, deep learning, AES

1 INTRODUCTION

Side-channel attacks extract secrets from physical devices by exploiting vulnerabilities in the implementation via non-primary, side channels [1, 2]. By finding the correlation between the physical measurements (power consumption, electromagnetic (EM) emissions, timing) taken at various points during the computation and the internal state of the processing device, the attacker can deduce the internal state of the device and then extract the related sensitive information, e.g. the secret key [3–5]. Different types of side channels have been successfully exploited to break physical implementations of cryptographic primitives [6, 7], algorithms [8–10] and protocols [11], steal intellectual property from FPGAs [12], and reverse-engineer neural networks [13].

Table 1: Summary of previous far field EM side-channel attacks on nRF52832 implementation of AES-128.

Previous attacks	# Traces to extract the key	Distance to the device	# Encryption repetitions
[14]	52K	1m	500
[15]	5K	15m	1K
[16]	10K	15m	1
This work	341	15m	1

Previous work. Attacks based on power or near filed EM side channels [17–19] require that the attacker has a direct physical access to the device under attack to perform measurements. Recently several side-channel attacks on Advanced Encryption Standard (AES) based on far field EM emissions have been presented [14–16]. Far field EM emissions can be measured on a distance from the device under attack and are a new type of side-channel which has been explored much less compared to timing, power and near field EM. Camurati et al. [14] presented the first template attack on AES-128 based on far field EMs. They observed that side channels from an AES implementation on a mixed-signal chip may unintentionally couple with the signal transmitted by the on-chip antenna. By analyzing the transmitted signal, it may be possible to recover the AES key. In [14], the AES-128 key is recovered from 52K traces captured at 1m distance to the target device in an office environment. Each trace is obtained by averaging out 500 measurements of the same encryption. In the follow up template attack [15], the AES-128 key is recovered from 5K traces captured at 15 m to the target device in an office environment using 1K repetitions of the same encryption and key enumeration up to 2^{23}. Finally, in [16] a deep learning-based attack is presented and further improved the attack efficiency. They train neural networks on traces captured at different distances from the profiling devices, and show that the model can recover the key from about 400 traces captured at 15 m distance to the target device with repeating each encryption 1K times. However, in a real attack scenario, it is impossible to capture traces with repeating each encryption more than once. For this case, [16] requires 10K traces captured at 15 m distance to the target device to recover a subkey, which is not efficient enough. In this paper, we train neural networks on traces captured through a coaxial cable.

*Corresponding author.

CPSS '21, June 7, 2021, Virtual Event, Hong Kong

ACM ISBN 978-1-4503-8402-5/21/06…$15.00
https://doi.org/10.1145/3457339.3457982

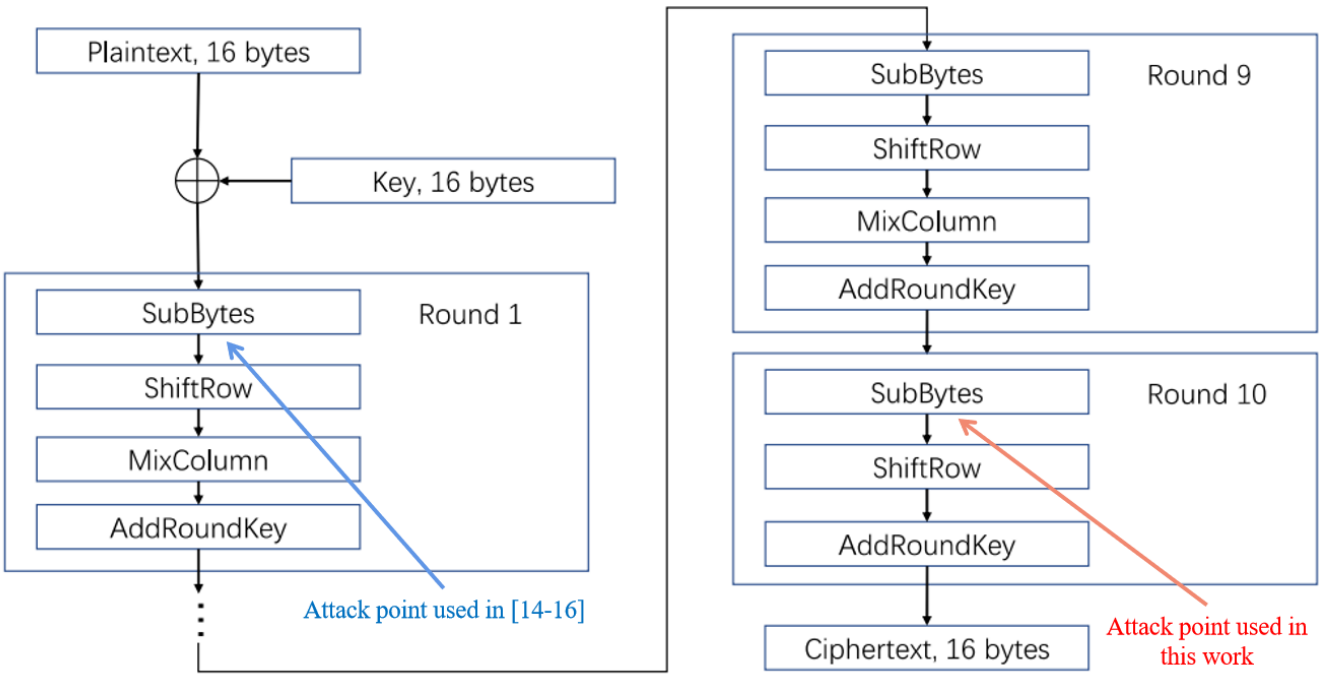


Figure 1: AES-128 algorithm.

Table. 1 shows a summary of previous profiled attacks based on far field EM emissions against nRF52832 implementations of AES-128.

Our contributions. Our main contribution is an observation that neural networks trained on traces captured through a coaxial cable from profiling devices with some noise added perform considerably better than networks trained on traces captured at a distance from profiling devices, as in [14–16]. Our best neural network model can extract the key from 341 traces on average captured at 15 m distance from the device under attack without repeating each encryption more than once. This is a 29-fold improvement over the attack presented in [16] which requires 9954 traces for the same conditions.

Paper organization. The rest of the paper is organized as follows. Section 2 provides background information on AES algorithm, deep learning side-channel attacks, and EM emissions. Section 3 defines the adversary model. Section 4 presents the equipment and the methods used to acquire and pre-process traces. Sections 5 and 6 describe the profiling and the attack stages, respectively. Section 7 summarizes the experimental results. Section 8 concludes this paper and discusses open problems.

2 BACKGROUND

This section provides background information on AES-128, side-channel attacks, and the usage of EM emissions as a side channel.

2.1 AES-128 algorithm

The AES [20] is a symmetric encryption algorithm standardized by NIST in FIPS 197 and included in ISO/IEC 18033-3. It takes a 128-bit block of plaintext $\mathcal{P}$ and an n-bit key K, $n = \{128, 192, 256\}$, as input and computes a 128-bit block of ciphertext C as output. In this paper, we use the AES with the key size $n = 128$, called AES-128.

The flow of the AES-128 algorithm is illustrated in Fig. 1. AES-128 performs 10 rounds of encryption. Each round except the last one repeats the following four steps: non-linear substitution, *SubBytes*, transposition of rows, *ShiftRows*, mixing of columns, *MixColumns*, and round key addition, *AddRoundKey*, but uses a different round key, RKi, $i \in \{1, \ldots, 10\}$, derived from the original key K. The last round does not mix columns.

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.

2.2 Side-channel attacks

The aim of a side-channel attack is to retrieve some secret, e.g. the encryption key $K \in \mathcal{K}$ of a cipher, where $\mathcal{K}$ is the set of all possible keys. To recover K, the attacker uses a set of known input data $\mathcal{X}$ (e.g. the plaintext, ciphertext) and a set of physical measurements $\mathcal{T}$ (e.g. power consumption, EM emissions, timing). The measurements $\mathcal{T}$ are analyzed to find the correlation with the key. The analysis is usually done by partitioning K into b-bit *subkeys*, K_k, and recovering each subkey independently for $k \in \{1, 2, \ldots, \frac{|K|}{b}\}$.

During the analysis, the attacker computes $\frac{|K|}{b}$ vectors of probabilities p_k whose elements, $p_{k,j}$, represent the probability that the subkey $K_k = j$ is the correct subkey, for $j \in \{0, 1, \ldots, 2^b - 1\}$. To guide the selection of the right candidate, the estimation metrics defined in the next subsection can be used.

2.2.1 Estimation metrics.

Rank. The *rank* of a key $K \in \mathcal{K}$, $R(K)$, is the number of keys with a higher probability than the one of K

$$R(K) = |\{K' \in \mathcal{K} : Pr[K|\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{T}}] < Pr[K'|\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{T}}]\}|.$$

Guessing entropy. The *Guessing Entropy*, GE, is the expected rank among all possible keys

$$GE = \mathop{\mathbb{E}}_{K \in \mathcal{K}}(R(K)).$$

GE provides a useful estimation of the number of key candidates required to be evaluated for a successful attack. If b-bit subkeys of K are recovered independently, for $k \in \{1, 2, \ldots, \frac{|K|}{b}\}$, then the entropy is guessed for each subkey K_k separately and *Partial Guessing Entropy*, PGE, rather that GE is used as an estimation metric [21].

2.2.2 Profiled side-channel attacks.

The main idea of a profiled attack is to create a leakage "profile" of a target device *in advance*, in order to decrease the amount of side-channel information necessary to break the device under attack during the actual attack. This may be important because, for example, physical access to the device under attack is only possible within a short time, or because the device frequently changes its key.

A profiled side-channel attack is performed in two stages: profiling and attack. Profiling can be done by creating a template [14, 22, 23], or a model, e.g. an artificial neural network [24–29].

To create a profile using the deep learning-based method, a neural network is trained to learn leakage of the target device for all possible values of the sensitive variable. The training is done using a large number of traces captured from the profiling device(s) which are labelled according to the selected leakage model (e.g. Hamming weight, Hamming distance, identity). Then, at the attack stage, the trained network is used to classify the traces captured from the device under attack.

2.3 EM emissions as a side channel

EM emissions can be classified into direct and indirect ones. These two types of emissions are caused by different phenomena and have different transmission properties.

During the execution of a cryptographic algorithm in the Crypto block, the logic components in the Crypto block change their states synchronously, according to the system clock. A sharp current change in the logic components leads to *direct* EM emissions. These emissions have high frequency components which can typically be detected by a near-field probe. Decapsulation may be required to acquire good direct EM emissions in some cases [8, 30].

Indirect EM emissions arise from the coupling effect between various components on chip. For example, a square wave noise is created by the frequently switching clock signal from the CPU core. Side channels from the Crypto block get modulated by this square wave. Due to the substrate coupling [31], the resulting signal couples with the digital signal representing the ciphertext and transfers through the Bus to the analog part where it is converted to an analog signal by the Digital-to-Analog Converter (DAC). The RF block and Voltage-Controlled Oscillator (VCO) modulate the analog signal to a high frequency defined by the wireless transmission protocol and transmit it through the antenna. For this reason, it is possible to detect indirect EM emissions at a much longer distance than direct EM emissions.

Following [14], we focus on the capacitive coupling which leads to Amplitude Modulation (AM) of side channels from the Crypto block.

3 ADVERSARY MODEL

The adversary can be anyone who has equipment for EM far field acquisition and expertise in side-channel analysis and deep learning. We also assume that:

(1) The adversary has at least one *profiling* device which is similar to the device under attack and runs the same implementation of the AES-128 algorithm.
(2) The adversary has a full control over the profiling device(s), e.g. knows the secret key, can apply chosen inputs, and measure side-channel signals.
(3) The adversary has an access to the device under attack within 15 m distance range to measure far field EM emissions during the execution of the AES-128 algorithm and eavesdrop on the ciphertext.

4 TRACE ACQUISITION AND PRE-PROCESSING

This section describes how traces containing indirect EM emissions were captured and pre-processed.

4.1 Measurement Setup

We use the same equipment as in [14–16]. The device under attack is a Bluetooth 5-supported nRF52832 with a data transmission rate of 2Mbps. The nRF52832 contains an ARM Cortex M4 CPU running at 64 MHz. It is mounted on the Nordic nRF52 DK board, a development kit suitable for implementing custom programs for the nRF52 series. In nRF52832, the C implementation of *TinyAES* from [32] with a 128-bit key is used. The nRF52832 periodically runs AES encryption with fixed 128-bits key and random plaintexts.

An Ettus Research USRP N210 software defined radio is used as a receiver. Its center frequency is set to $2f_{clock} + f_{\text{Bluetooth}} = 2.528\text{GHz}$, where $f_{\text{Bluetooth}} = 2.4\text{GHz}$ is the Bluetooth channel center frequency and $f_{clock} = 64\text{MHz}$ is the frequency of the CPU clock. To determine the center frequency of the receiver, we used the same way as in [14–16]. The sampling frequncy is set to 5 MHz. The signal is received using a grid parabolic antenna TL-ANT2424B with 24dBi gain.

4.2 Trace Acquisition

In our experiments, as in [14, 16], the transmitter sends the data continuously. The data sent by the RF block consists of two major parts: (1) the ciphertext with the central frequency at 2.4Ghz, and (2) the modulated side channel leakage with the central frequency at 2.528Ghz. The ciphertext is generated by AES and the side channel leakage (EM emission) is produced by switching activity of the logic components in the Crypto block. The AES encryptions are found in periodic blocks in the received data stream, as shown at

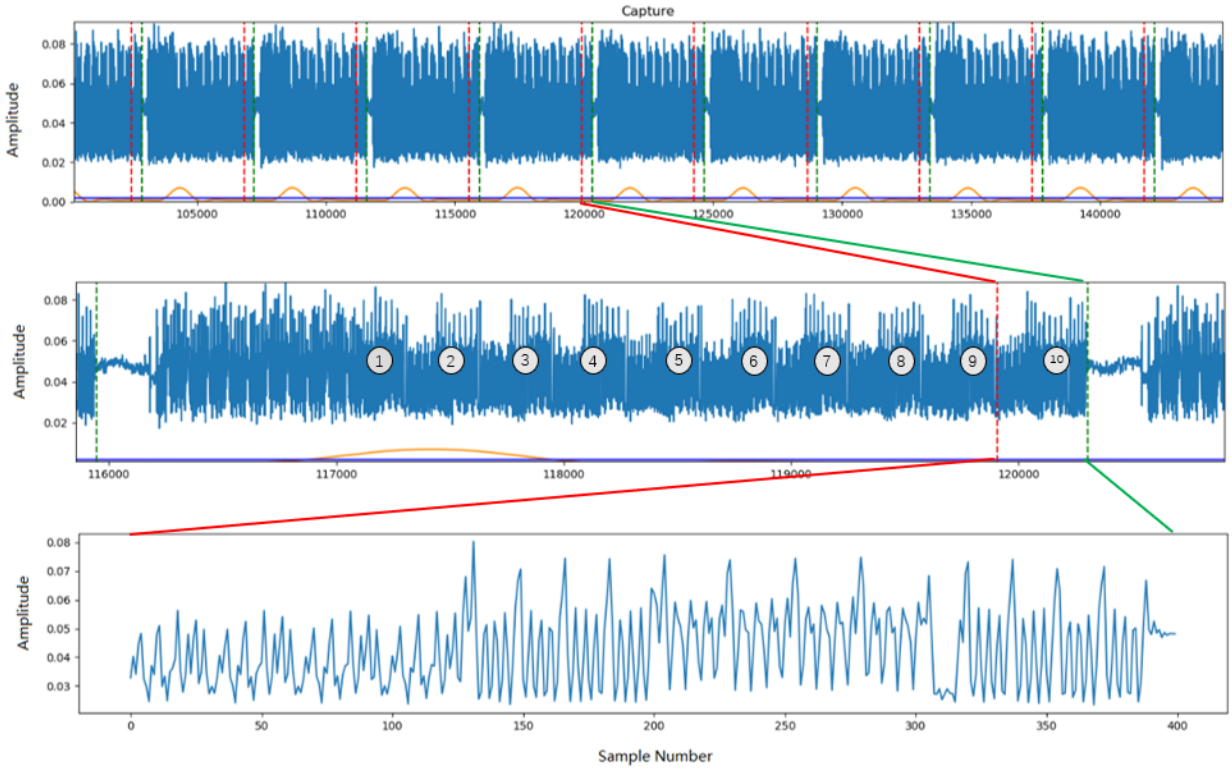


Figure 2: Locating the last round of AES in the received data stream.

the top of Fig. 2. If we zoom in one block, we can clearly see the ten encryption rounds of AES-128 (see the middle picture of Fig. 2).

In order to find an approximate start of each encryption, we use a trigger signal, which is generated from the received signal. The received signal contains AM modulated I/Q samples. To extract the trigger, we post-process the I/Q samples by taking the absolute value, then use a 5-order Butterworth band pass filter with 1.85MHz and 1.95MHz lower and upper band frequencies, respectively, and the 5-order Butterworth low pass filter with 5KHz cutoff frequency.

The orange line at the bottom of the top picture in Fig. 2 shows a trigger signal filtered from the AM demodulated I/Q samples. The blue horizontal line at the bottom of the top picture in Fig. 2 shows the average value of the trigger signal. Each encryption starts approximately at the intersection of the blue line with the rising edge of the orange line.

After determining the approximate start of each encryption, we add an offset to the intersection point in order to locate the last round. In the presented attack, the value of the S-box output in the last round is used as a label for traces, see Fig. 1.

At the bottom picture of Fig. 2, the dashed red and green lines show the beginning and the end of a 400 data point interval, respectively, containing *MixColumns* and *AddRoundKey* operations of the 9th round and the complete last round of AES-128. One can see the two identical patterns within approx. 130-200 points and 320-390 points corresponding to the two executions of *AddRoundKey* in the 9th and 10th rounds, respectively.

4.3 Trace pre-processing

After trace acquisition, we pre-process them by aligning, scaling and averaging.

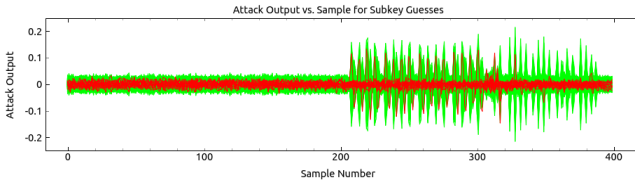
The alignment helps us to synchronize the traces precisely. The scaling is necessary because, for training, we use traces captured through a coaxial cable and, for testing, we use traces captured at a distance from the device. Note that the coaxial cable connects the transmitter and the receiver directly to receive the EM emission. This emission is an RF signal sent from the RF block on the chip. A coaxial cable can transmit signals that oscillate at high RF frequencies without radiating them outside. Since the amplitude of the received signal is proportional to the inverse of the distance to the device, the amplitudes of trace captured through a cable and at the distance are not within the same range.

Following [16], we use *min-max scaling* [33] to map the amplitude of all traces to the interval [0,1].

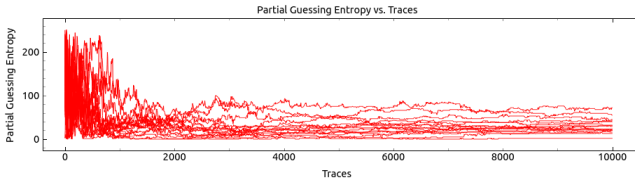
In addition, traces for training are captured by repeating the same encryption 100 times and averaging out the resulting traces. So, each training trace is the average of 100 measurements of the same encryption. Traces for testing are captured without repeating the same encryption more than once (except for one experiment).

4.4 Correlation analysis

To show the reader how non-profiling attacks such as Correlation Power Analysis (CPA) perform on far field EM side-channels and

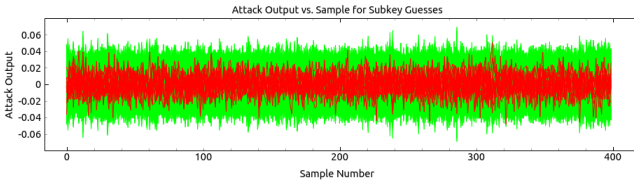


(a) Correlation coefficients for all subkeys.

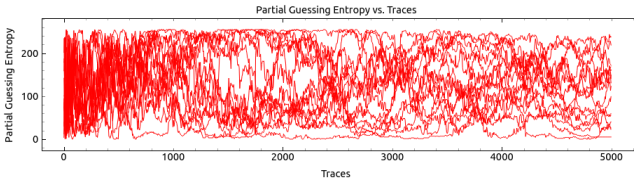


(b) PGE vs number of traces for all subkeys.

Figure 3: CPA results for 10K traces captured through a coaxial cable with 100 repetitions.



(a) Correlation coefficients for all subkeys.



(b) PGE vs number of traces for all subkeys.

Figure 4: CPA results for 5K traces captured at 15 m distance without repetitions.

how the number of repetitions affect the quality of traces, in this section we present CPA results for traces captured with 100 repetitions and without repetitions from an nRF52832 device using the equipment and method presented in the previous section.

The Hamming weight of the S-box output in the last round of AES is used as a leakage model for the CPA. Fig. 3(a) shows the correlation results for all 16 subkeys for 10K traces captured with 100 repetitions. The red and green lines represent the correlation coefficients of the correct subkey and the rest of subkey guesses, respectively. Note 16 peaks in the interval between approx. 200 and 310 points corresponding to S-box evaluations.

As we can see from the PGE plots in Fig. 3(b), the CPA cannot recover any subkey within 10K traces without the key enumeration. The minimum rank is 2 and the maximum is 72.

Obviously, the CPA on traces captured on a distance from the device under attack, and/or captured with less repetitions, is even more difficult. As an example, see Fig. 4 which shows CPA results for 5K traces captured at 15 m distance from the device without repetitions.

5 PROFILING STAGE

This section describes how we train neural networks at the profiling stage.

Let $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_{|\mathcal{T}|}\}$, where $\mathcal{T}_i \in \mathbb{R}^m$, for $i \in \{1, \ldots, |\mathcal{T}|\}$, be a set of traces representing the computation of the last round of AES which are captured from the profiling device(s) for randomly generated plaintexts $\mathcal{P}_i \in \{0,1\}^{128}$ and a fixed key $K \in \{0,1\}^{128}$. Let $C_i \in \{0,1\}^{128}$ be the ciphertext which is generated when the trace $\mathcal{T}_i$ is captured. Let $C_{i,k}$ denote the kth byte of the ciphertext C_i, $k \in \{0, 1, \ldots, 15\}$. Let $\mathbb{I} = \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$.

5.1 Labeling strategy

In theory, it is possible to train a single network capable of recovering all subkeys, as demonstrated in [34]. However, network trained for a fixed subkey typically achieves a higher classification accuracy [35]. Taking this into account, we train a separate network for each subkey K_k, $k \in \{0, 1, \ldots, 15\}$.

To train a network for a subkey K_k, each trace $\mathcal{T}_i \in \mathcal{T}$, $i \in \{1, \ldots, |\mathcal{T}|\}$, is assigned a label, $l_k(\mathcal{T}_i)$, computed as

$$l_k(\mathcal{T}_i) = C_{i,k} \oplus RK10_k,$$

where $RK10_k$ is the kth byte of the last round key $RK10$. It is easy to see that the value of $l_k(\mathcal{T}_i)$ is equal to the value of the S-box output in the last round when the kth byte of C_i is computed.

The resulting labeled set of traces is used to train a neural network $\mathcal{N}_k : \mathbb{R}^m \to \mathbb{I}^{256}$ which maps a trace $\mathcal{T}_i \in \mathbb{R}^m$ into a *score* vector $S_{i,k} = \mathcal{N}_k(\mathcal{T}_i) \in \mathbb{I}^{256}$ whose elements $s_{i,k,j}$ represent the probability that the S-box output in the last round is equal to $j \in \{0, 1, \ldots, 255\}$ when the kth byte of C_i is computed:

$$s_{i,k,j} = \Pr(C_{i,k} \oplus RK10_k = j).$$

5.2 Training details

We use the same strategy as in [16] for training of neural networks. The deep-learning model used in our experiments is Convolutional Neural Network (CNN), which has been successfully applied to bypass trace misalignment and to overcome jitter-based countermeasures [25]. Layer structures of CNN classifiers are shown in Table 2 and Table 8. Categorical cross-entropy loss is used to quantify the classification error of the network. To minimize the loss, the gradient of the loss with respect to the score is computed and back-propagated through the network to tune its internal parameters according to the RMSprop optimizer (which is one of the advanced extensions of the Stochastic Gradient Decent (SGD) algorithm [36]) with the learning rate 0.0001 and no learning rate decay. The training is carried out for a maximum of 100 epochs with batch size 128. At each iteration, the model is stored instead of being overwritten. This strategy gives us t models $\mathcal{N}_k^e$ for different epochs $e = \{1, 2, \ldots, 100\}$ at the end of training, for each k.

Table 2: The architecture of the *CNN* from [16].

Layer (Type)	Output Shape	Parameter #
Input (Dense)	(None, 110, 1)	0
Conv 1 (Conv1D)	(None, 110, 4)	16
AveragePooling 1	(None, 109, 4)	0
Conv 2 (Conv1D)	(None, 109, 8)	104
AveragePooling 2	(None, 108, 8)	0
Conv 3 (Conv1D)	(None, 108, 16)	400
AveragePooling 3	(None, 107, 16)	0
Conv 4 (Conv1D)	(None, 107, 32)	1568
AveragePooling 4	(None, 106, 32)	0
Flatten 1 (Flatten)	(None, 3392)	0
Dense1 (Dense)	(None, 200)	678600
Dense2 (Dense)	(None, 200)	40200
Output (Dense)	(None, 256)	51456

Total Parameters: 772,344

6 ATTACK STAGE

This section describes how we test neural networks.

Let $\hat{\boldsymbol{\mathcal{T}}} = \{\hat{\mathcal{T}}_1, \ldots, \hat{\mathcal{T}}_{|\hat{\boldsymbol{\mathcal{T}}}|}\}$, where $\mathcal{T}_i \in \mathbb{R}^m$, for $i \in \{1, \ldots, |\hat{\boldsymbol{\mathcal{T}}}|\}$, be an ordered set of traces representing the computation of the last round of AES which are captured from the device(s) under attack. The same m-point segment of the trace as the one used for the profiling stage is selected.

At the attack stage, the trained networks $\mathcal{N}_k^e$ are used to recover the subkeys K_k from $\hat{\boldsymbol{\mathcal{T}}}$ for every $k \in \{0, 1, \ldots, 15\}$ as follows. The traces $\mathcal{T}_i \in \hat{\boldsymbol{\mathcal{T}}}$ are applied to $\mathcal{N}_k^e$ in order and the most likely label $\tilde{l}_k$ for $\mathcal{T}_i$ is determined among all candidate labels as

$$\tilde{l}_k = \underset{j \in \{0,1,\ldots,255\}}{\arg\max} \left(\prod_{p=1}^{i} s_{p,k,j}\right), \tag{1}$$

where $s_{p,k,j}$ is the jth element of the score vector $S_{p,k} = \mathcal{N}_k(\mathcal{T}_p)$ of a trace $\mathcal{T}_p \in \hat{\boldsymbol{\mathcal{T}}}$ which precedes $\mathcal{T}_i$ in $\hat{\boldsymbol{\mathcal{T}}}$. Once $\tilde{l}_k = l_k(\mathcal{T}_i)$, the classification is successful.

To verify if $\tilde{l}_k = l_k(\mathcal{T}_i)$ we permute $\hat{\boldsymbol{\mathcal{T}}}$ r times to get the test sets $\hat{\boldsymbol{\mathcal{T}}}_1, \ldots, \hat{\boldsymbol{\mathcal{T}}}_r$ and apply them to $\mathcal{N}_k^e$. The point where the rank of K_k for $\mathcal{N}_k^e$ tested on $\hat{\boldsymbol{\mathcal{T}}}_j$, $R_j^e(K_k)$, reaches 0 in the majority of test sets $\hat{\boldsymbol{\mathcal{T}}}_1, \ldots, \hat{\boldsymbol{\mathcal{T}}}_r$ is the termination point. This is a different termination condition from the one used in [16]. In [16] the condition $\overline{R}(K_k) \leq 0.5$ is used, where $\overline{R}(K_k)$ is the average rank of K_k for $\mathcal{N}_k^e$ tested on $\hat{\boldsymbol{\mathcal{T}}}_1, \ldots, \hat{\boldsymbol{\mathcal{T}}}_r$:

$$\overline{R}(K_k) = \frac{\sum_{j=1}^{r} R_j^e(K_k)}{r}.$$

Note that, $\overline{R}(K_k) \leq 0.5$ implies that $R_j^e(K_k) = 0$ in the majority of test sets. However, the opposite is not true, i.e. the rank may reach 0 in the majority test sets before $\overline{R}(K_k)$ reaches 0.5. This may happen, for example, if $R_j^e(K_k) = 0$ for $j = \{1, 2, \ldots, r-1\}$ and $R_r^e(K_k) = r/2 + 1$, for an even r. Therefore checking if $R_j^e(K_k) = 0$ in the majority of test sets always results in either an earlier, or the same termination point as the condition $\overline{R}(K_k) \leq 0.5$.

Once the condition $\tilde{l} = l(\mathcal{T}_i)$ is satisfied and the correct label $l(\mathcal{T}_i)$ is found for some $i \in \{1, \ldots, |\hat{\boldsymbol{\mathcal{T}}}|\}$, the kth subkey of the 10th round key $RK10_k$ is recovered as

$$RK10_k = l_k(\mathcal{T}_i) \oplus C_{i,k}.$$

After recovering all subkeys of $RK10$, the original key K is computed from $RK10$ be reversing the AES-128 key expansion algorithm [20].

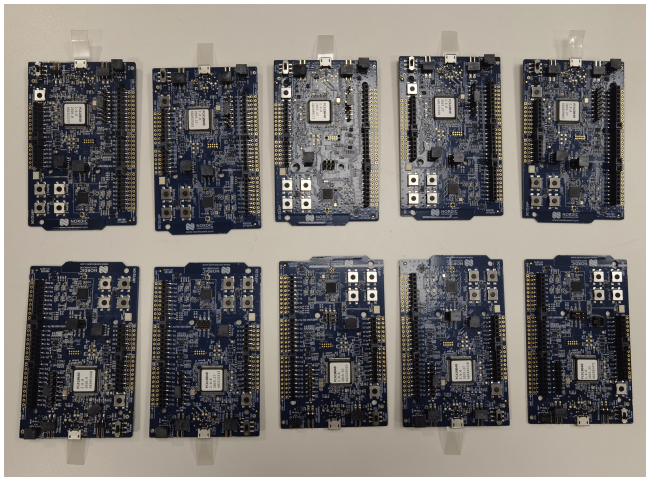


Figure 5: Ten nRF52832 devices used in the experiments.

Table 3: Summary of the profiling trace set $\boldsymbol{\mathcal{T}}$ in the first experiment. Each trace in $\boldsymbol{\mathcal{T}}$ is the average of 100 measurements of the same encryption.

Distance to device	Profiling device				
	D_1	D_2	D_3	D_4	D_5
cable	20K	20K	20K	20K	20K

7 EXPERIMENTAL RESULTS

This section presents the results of our experiments[1]. Ten identical nRF52832 devices shown in Fig. 5 are used. The devices $D_1 - D_5$ are used for profiling and the devices $D_6 - D_{10}$ for attack. All attacks are carried out in a corridor in an office building at 15 m distance from the device under attack. In all attacks, the subkey K_6 is recovered. The choice of the subkey does not seem to affect the average results. We have chosen K_6 because in the CPA presented in section 4.4 K_6 has the largest PGE, 72 (see Fig. 3).

7.1 Training on traces captured through a cable

The aim of this experiment is to investigate if a neural network trained on traces captured through a coaxial cable can classify with a high accuracy traces captured at a distance from the device.

At the profiling stage, we train CNNs $\mathcal{N}_6^e$ with the same architecture as in [16] (see Table 2) for 100 epochs, $e = \{1, \ldots, 100\}$. We use 100K training set $\boldsymbol{\mathcal{T}}$ composed as a union of 20K sets from five different devices captured through a coaxial cable (see Table 3). We

[1] All our code and traces will be publicly available at https://github.com/NymeriaWang/Advanced-Far-Field-EM-Side-Channel-Attack-on-AES.

Table 4: Average number of traces required to recover subkey K_6 by CNN $\mathcal{N}_6^e$ trained on the first and last rounds from traces captured without repetitions at 15 m distance from the device under attack (for 100 tests).

Device under attack	Last round		First round	
	Average number of traces	Best epochs e	Average number of traces	Best epochs e
D_6	503	33	1208	100
D_7	943	40	> 2000	for all e
D_8	260	89	1467	34
D_9	504	40	> 2000	for all e
D_{10}	187	30	1206	56
average	479.4		> 1576	

Table 5: Average number of traces required to recover subkey K_6 by CNN $\mathcal{N}_6^{41}$ from traces captured at 15 m distance from the device under attack (for 100 tests). Each trace is the average of N measurements of the same encryption.

Device under attack	$N = 1$	$N = 10$	$N = 100$
D_6	615	58	12
D_7	743	62	11
D_8	387	53	14
D_9	594	90	21
D_{10}	214	34	8
average	510.6	59.4	13.2

Table 6: Average number of traces required to recover subkey K_6 by CNN $\mathcal{N}_6^{41}$ from traces captured without repetitions at 15 m distance from the device under attack for different termination criteria (for 100 tests).

Device under attack	Majority of $R_j^{41}(K_6) = 0$ for $j \in \{1, \ldots, 100\}$	$\overline{R}(K_6) \leq 0.5$
D_6	615	775
D_7	743	896
D_8	387	524
D_9	594	720
D_{10}	214	349
average	510.6	652.8

verify that the receiver is not overloaded by visualizing the I/Q samples without cut-off for both real and imaginary parts. Each trace in $\mathcal{T}$ is the average of 100 measurements of the same encryption. The 110-point segment comprising each trace is shown at the bottom of Fig. 10.

At the attack stage, we test the trained networks $\mathcal{N}_k^e$ for $e = \{20, \ldots, 100\}$ on the trace sets $\hat{\mathcal{T}}$ of size 2K captured from $D_6 - D_{10}$ at the distance 15 m to the device without repeating the same encryption more than once. We use $e = 20$ as a starting point because the models $\mathcal{N}_k^e$ for $e < 20$ are obviously underfitted. Each trace of $\hat{\mathcal{T}}$ represents the same 110-point segment (shown in Fig. 10) as in the traces of the training set $\mathcal{T}$. The average numbers of traces required to recover the subkey K_6 from devices $D_6 - D_{10}$ are listed in the column 2 of Table 5. To compute the average number of traces, for each fixed $e = \{20, \ldots, 100\}$, we permute the trace set $\hat{\mathcal{T}}$ 100 times and calculate the point where the rank $R_j^e(K_6)$ reaches 0 in the majority of test sets $\hat{\mathcal{T}}_j$, $j \in \{1, \ldots, 100\}$. The numbers listed in the column 2 of Table 5 are for the best number of epochs, $e = 41$.

From Table 5 we can see that $\mathcal{N}_6^{41}$ can recover the subkey K_6 from 510.6 traces on average. This is a 19.5-fold improvement over the attack presented in [16] which requires 9954 traces on average for the same conditions. In [16] networks are trained on traces captured from five different devices at five different distances to the device, including the coaxial cable. This shows that training on traces captured only through a coaxial cable is more advantageous.

We also check how the results change if traces from the device under attack are captured with multiple repetitions of the same encryption. We do not think that such an attack setting is realistic. However, it is interesting to compare our results to the results of [14–16] which use multiple repetitions (from 100 to 1K).

We use the CNN $\mathcal{N}_6^{41}$ to recover the subkey K_6 from the trace sets of size 1K captured from $D_6 - D_{10}$ at the distance 15 m to the device in which the same encryption is repeated $N = 10$ and 100 times. Column 3 and 4 of Table 5 summarize the results. We can see that the case of $N = 10$ repetitions reduces the average number of traces 8.6 times compared to the case without repetitions. For $N = 100$ repetitions, the reduction is 38.7 times.

In the attack presented in [16], the best result for traces captured with 100 repetitions under the same conditions is 2946 on average, which is 223 times larger than the average result 13.2 in Table 5. For traces captured with 1K repetitions under the same conditions, the attack in [16] requires 367 traces on average. The template attack presented in [15], requires 5K traces with 1K repetitions and key enumeration up to 2^{23} to recover the key at 15 m distance. The template attack presented in [14] requires 52K traces with 500 repetitions to recover the key at 1 m distance. This shows that training on traces captured through a coaxial cable is also considerably more advantageous for the case when traces from the device under attack are captured with multiple repetitions.

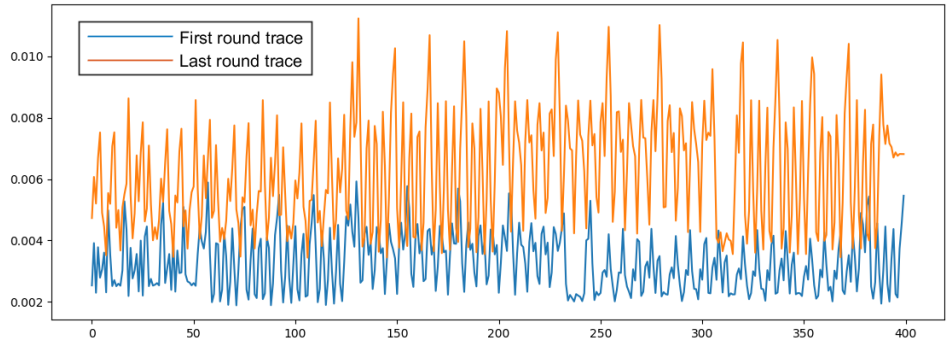


Figure 6: The first and the last round traces captured at 15 m to device (without scaling, averaged over 10K traces).

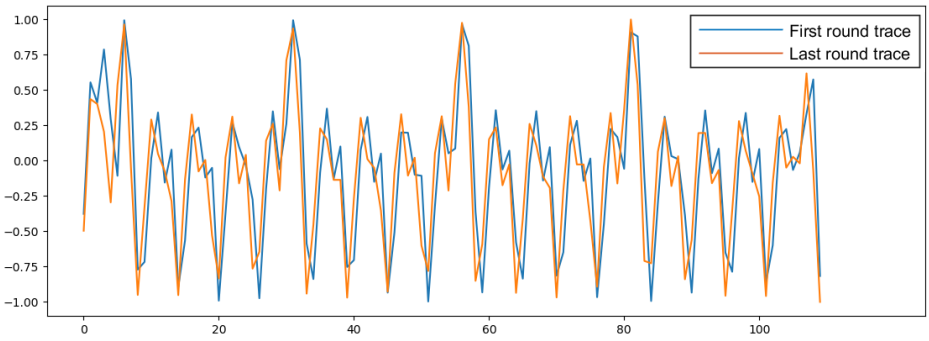


Figure 7: The first and the last round traces captured by cable (scaled to [-1,1], averaged over 10K traces).

7.2 Evaluating other changes in training strategy

From the previous experiment we can conclude that one of the changes which contributes to the improvement of the results of [16] is training on traces captured through a coaxial cable. The aim of this experiment is to quantitatively evaluate how other changes:

(1) changing the attack point to the last round, and
(2) changing the termination condition

contribute to the improvement.

7.2.1 First round vs. last round. First, we create a new 100K training set in which each trace represents the 110-point segment containing 16 S-box evaluations in the first round (same as in [16]). The trace set is constructed as in Table 3 and labeled by the value of the S-box output in the first round. We use this training set to train CNNs with the same architecture as in Table 2. The CNNs are trained in the same way as in the experiment 1, for 100 epochs $e \in \{1, \ldots, 100\}$.

We test the trained networks on the trace sets of size 2K captured without repetitions from $D_6 - D_{10}$ at 15 m distance from the device. Table 4 lists the results and the best number of epochs e for each case. From Table 4 we can see that changing the attack point to the last round contributes at least a three-fold improvement.

Next we compare the first and the last round traces in order to analyse why the number of traces required for the attack in the two cases differs. Fig. 6 shows the plots obtained by averaging 10K 400-point traces captured at 15 m to device for random plaintexts. The blue plot represents computations in the 1st round and the orange one - in the last round. No scaling is applied to traces. It is evident that the side-channel signal in the last round's traces is considerably stronger. Since traces for the first and the last rounds were captured in different days (several months apart), possibly the positioning of equipment and environmental conditions were different in the two cases.

Fig. 7 shows the plots obtained by averaging 10K traces captured by cable for random plaintexts. The blue plot represents 16 Sbox computations in the 1st round and the orange one - in the last round. All traces are scaled to the interval [-1,1]. We can see that there are some differences in the shape of traces, but they are not significant.

Fig. 8 and Fig. 9 illustrate how well traces captured at 15 m to device fit traces captured by cable (after [-1,1] scaling) for the first and the last round cases, respectively. In both figures, the blue/orange plot represents an average of 10K traces captured by cable/at 15 m to device for random plaintexts. The figures show that, for the last round, there is a closer fit. We explain it by the fact that the side-channel signal in the last round's traces captured at 15 m to device is stronger than the signal in the first round's traces captured at 15 m to device, as shown in Fig. 6. We believe that this is the reason why the last round-based attack requires less traces.

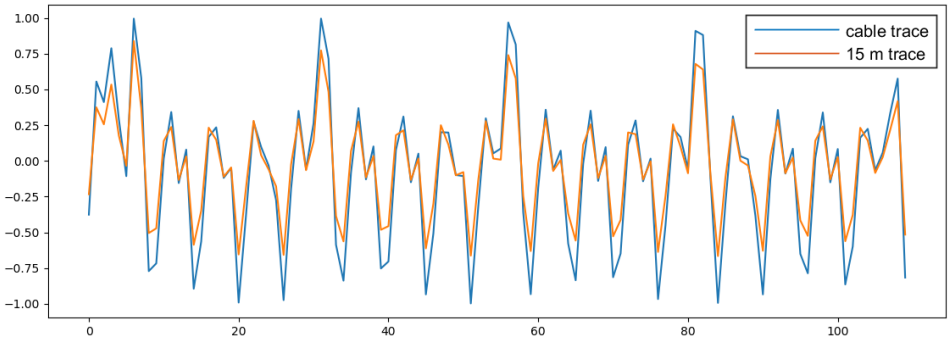


Figure 8: The first round traces captured at 15 m to device and by cable (scaled to [-1,1], averaged over 10K traces).

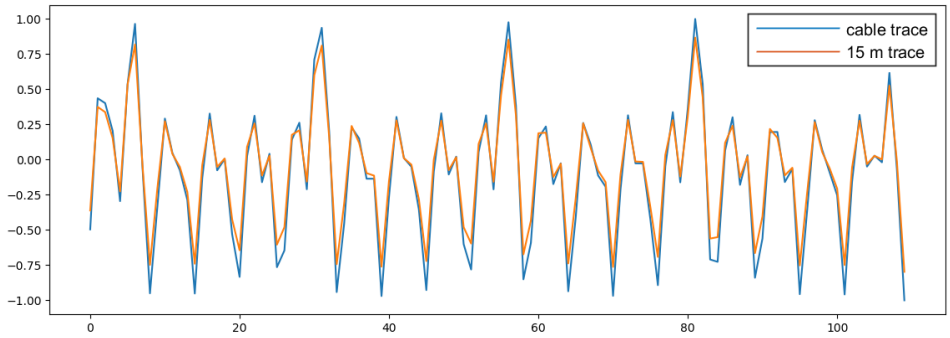


Figure 9: The last rounds traces captured at 15 m to device and by cable (scaled to [-1,1], averaged over 10K traces).

Table 7: Average number of traces required to recover subkey K_6 by CNN $\mathcal{N}_6^{41}$ from traces captured without repetitions at 15 m distance from the device under attack for the case when Gaussian white noise with mean μ and standard deviations σ is added to the training set (for 100 tests).

Device under attack	$\mu = 0$ $\sigma = 0.0065$	$\mu = 0$ $\sigma = 0.01$
D_6	414	292
D_7	854	467
D_8	309	256
D_9	599	384
D_{10}	225	306
average	480.2	341.0

Table 8: 400-input CNN architecture.

Layer (Type)	Output Shape	Parameter #
Input (Dense)	(None, 400, 1)	0
Conv 1 (Conv1D)	(None, 400, 8)	128
AveragePooling 1	(None, 399, 8)	0
Conv 2 (Conv1D)	(None, 399, 16)	656
AveragePooling 2	(None, 398, 16)	0
Conv 3 (Conv1D)	(None, 398, 32)	2592
AveragePooling 3	(None, 397, 32)	0
Conv 4 (Conv1D)	(None, 397, 64)	10304
AveragePooling 4	(None, 396, 64)	0
Flatten 1 (Flatten)	(None, 25344)	0
Dense1 (Dense)	(None, 300)	7603500
Output (Dense)	(None, 256)	77056

Total Parameters: 7,694,236

7.2.2 Termination conditions. Second, we change the termination condition to the one in [16], namely checking if the average rank reached 0.5. Table 6 shows the results for the CNN $\mathcal{N}_6^{41}$. The second column of Table 6 is a copy the second column of Table 5. From Table 6 we can see that the average results for the termination condition $\overline{R}(K_6) \leq 0.5$ are 28% worse than the ones for the majority condition.

From the experiments 1 and 2 we can conclude that training on traces captured through a coaxial cable contributes most to the improvement over the results of [16]. The second contributor is the stronger side-channel signal in the last round's traces, and the third - the new termination condition.

7.3 Adding noise to training traces

In this experiment, we investigate if the addition of noise to training traces captured through a coaxial cable can further improve the classification accuracy. In this experiment, we add a small, controlled amount of additive noise to the training traces with $N = 100$

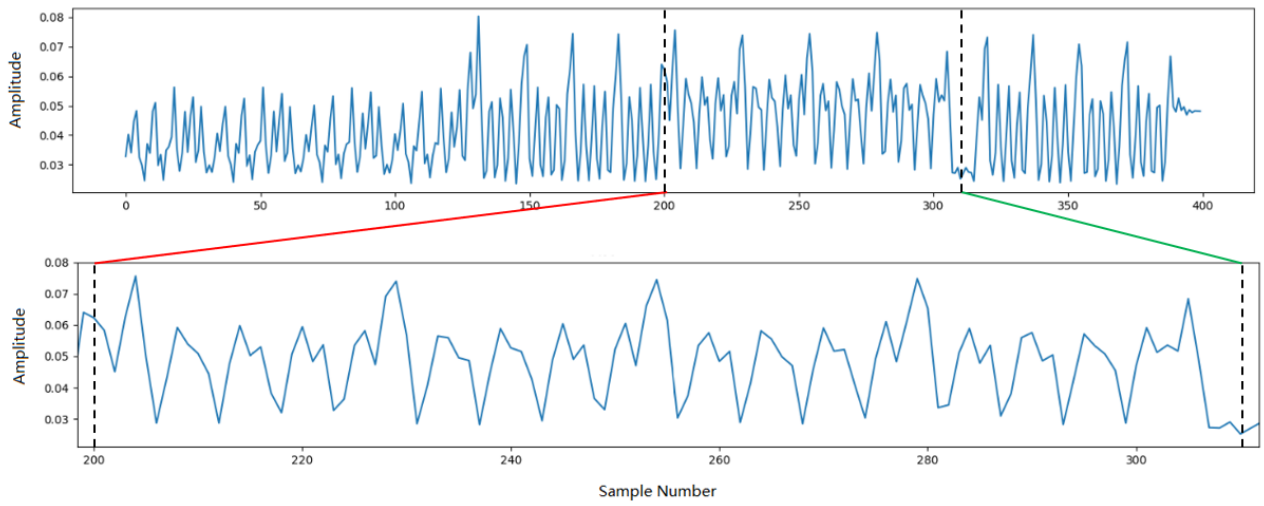


Figure 10: The 110-point segment used in the experiment 1 and its location within the 400-point trace in Fig. 2.

Table 9: Average number of traces required to recover subkey K_6 by the 400-input CNN from Table 8 for $e = 67$ from traces captured at 15 m distance from the device under attack (for 100 tests). Each trace is the average of N measurements of the same encryption.

Device under attack	$N = 1$	$N = 10$	$N = 100$
D_6	542	52	9
D_7	808	49	9
D_8	534	50	11
D_9	627	65	16
D_{10}	545	23	8
average	611.2	47.8	10.6

Table 10: Average number of traces required to recover subkey K_6 by the 400-input CNN from Table 8 for $e = 67$ from traces captured without repetitions at 15 m distance from the device under attack for the case when Gaussian white noise with mean μ and standard deviations σ is added to the training set (for 100 tests).

Device under attack	$\mu = 0$ $\sigma = 0.0065$	$\mu = 0$ $\sigma = 0.01$
D_6	828	498
D_7	738	759
D_8	255	395
D_9	401	406
D_{10}	320	274
average	524.8	466.4

repetitions rather than reducing N. Since traces captured without repetitions are affected by the additive (white) noise as well as by the multiplicative noise (interference). Averaging minimizes the impact of both types of noise. By adding a small amount of additive noise, we seem to help CNN generalize better without exceeding the amount of noise it can handle.

We add to the training set $\mathcal{T}$ constructed as in Table 3 white Gaussian noise with mean $\mu = 0$ and standard deviation $\sigma = 0.0065$ and $\sigma = 0.01$. We use the resulting training set to train CNNs as in the experiment 1. Then we test the network $\mathcal{N}_6^{41}$ on the trace sets $\hat{\mathcal{T}}$ of size 1K captured from $D_6 - D_{10}$ at the distance 15 m from the device without repeating the same encryption more than once. Table 7 lists the results.

We can see from Table 7 that the addition of noise to the training set $\mathcal{T}$ further improves the ability of CNN to classify traces captured at the distance 15 m from the device under attack. The average numbers of traces for $\sigma = 0.0065$ and $\sigma = 0.01$ are 6% and 33% better, respectively, than the ones in column 2 of Table 5. The average number for $\sigma = 0.01$ is 29 times smaller that the average number of traces 9954 required in the attack of [16] for the same conditions.

7.4 Extending input size

The purpose of this experiment is to check if using CNNs with a larger input size can be beneficial. Such a strategy is suggested in some works, e.g. [10].

We performed experiments for CNNs with the input size 400 trained on 400-point traces shown at the top of Fig. 10, and also for CNNs with the input sizes 800 and 1200 trained on the same segment of AES encryption captured with doubled and tripled sampling frequency, respectively. None of these network types were better than the 110-input CNN in Table 2. In this section we show the results for the CNN with the input size 400 which is the best of three. Its architecture is listed in Table 8.

The 100K training set is constructed as in Table 3 and labeled by the value of the S-box output in the last round. First, we repeat the experiment 1. Table 9 lists the results for the best epoch, $e = 67$. We can see that the average result for $N = 1$ is 19% worse than the average result 510.6 in Table 5. It is interesting that the average results for $N = 10$ and 100 are 20% better than the results 59.4 and 13.2 in Table 5.

We also repeat the experiment 3. Table 10 lists the results for the best epoch, $e = 67$. We can see that the average results for $\sigma = 0.0065$ and $\sigma = 0.01$ are 9% and 37% worse than the corresponding results in Table 7.

8 CONCLUSION

We achieved a 20-fold improvement in the average number of traces required for a successful attack over the previous work [14–16]. Our experiments show that training on traces captured through a coaxial cable with some noise added is considerably more advantageous that training on traces captured at a distance from the device under attack. We also introduced a new termination condition based on checking if the rank reaches 0 in the majority of test sets. We quantitatively evaluated the contribution of all factors to the improvement.

Future work includes finding the best strategy for modeling noise added to the training traces and mounting similar attacks on devices supporting other wireless network protocols.

ACKNOWLEDGEMENT

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.

REFERENCES

[1] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Annual International Cryptology Conference*. Springer, 1996, pp. 104–113.
[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.
[3] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side—channel (s)," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2002, pp. 29–45.
[4] P. Kocher, R. Lee, G. McGraw, and A. Raghunathan, "Security as a new dimension in embedded system design," in *Proceedings of the 41st annual Design Automation Conference*, 2004, pp. 753–760.
[5] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer Science & Business Media, 2008, vol. 31.
[6] G. T. Becker, R. Kumar *et al.*, "Active and passive side-channel attacks on delay based PUF designs." *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 287, 2014.
[7] Y. Yu, M. Moraitis, and E. Dubrova, "Profiled deep learning side-channel attack on a protected arbiter PUF combined with bitstream modification," IACR Cryptology ePrint Archive, 2020/1031, 2020, https://eprint.iacr.org/2020/1031.
[8] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2001, pp. 251–261.
[9] R. Gilmore, N. Hanley, and M. O'Neill, "Neural network based attack on a masked implementation of aes," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 106–111.
[10] M. Brisfors, S. Forsmark, and E. Dubrova, "How deep learning helps compromising USIM," in *Proc. of the 19th Smart Card Research and Advanced Application Conference (CARDIS'2020)*, Nov. 2020.
[11] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1997, pp. 37–51.
[12] A. Moradi and T. Schneider, "Improved side-channel analysis attacks on Xilinx bitstream encryption of 5, 6, and 7 series," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2016, pp. 71–87.
[13] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 515–532.
[14] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming channels: When electromagnetic side channels meet radio transceivers," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 163–177.
[15] G. Camurati, A. Francillon, and F.-X. Standaert, "Understanding screaming channels: From a detailed analysis to improved attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 358–401, 2020.
[16] R. Wang, H. Wang, and E. Dubrova, "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, pp. 35–44.
[17] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Study of deep learning techniques for side-channel analysis and introduction to ASCAD database," *ANSSI, France & CEA, LETI, MINATEC Campus, France. Online verfügbar unter https://eprint. iacr. org/2018/053. pdf, zuletzt geprüft am*, vol. 22, p. 2018, 2018.
[18] H. Wang, M. Brisfors, S. Forsmark, and E. Dubrova, "How diversity affects deep-learning side-channel attacks," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*. IEEE, 2019, pp. 1–7.
[19] H. Wang and E. Dubrova, "Tandem deep learning side-channel attack against fpga implementation of AES." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 373, 2020.
[20] D. Joan and R. Vincent, "The design of Rijndael: AES the advanced encryption standard," in *Information Security and Cryptography*. springer, 2002.
[21] H. Pahlevanzadeh, J. Dofe, and Q. Yu, "Assessing CPA resistance of AES with different fault tolerance mechanisms," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2016, pp. 661–666.
[22] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Template attacks in principal subspaces," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2006, pp. 1–14.
[23] C. Hoffman, C. Gebotys, D. F. Aranha, M. Cortes, and G. Araujo, "Circumventing uniqueness of XOR arbiter PUFs," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019, pp. 222–229.
[24] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 3–26.
[25] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 45–68.
[26] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. Unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 148–179, 2019.
[27] H. Wang, "Side-channel analysis of AES based on deep learning," Master's thesis, School of Electrical Engineering and Computer Science, KTH, 2019.
[28] H. Wang and E. Dubrova, "Federated learning in side-channel analysis," in *The 23rd Annual International Conference on Information Security and Cryptology*. Springer, 2020, pp. 255–270.
[29] H. Wang, S. Forsmark, M. Brisfors, and E. Dubrova, "Multi-source training deep-learning side-channel attacks," in *2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL)*. IEEE, 2020, pp. 58–63.
[30] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *International Conference on Research in Smart Cards*. Springer, 2001, pp. 200–210.
[31] S. Bronckers, G. Van der Plas, and Y. Rolain, *Substrate noise coupling in analog/RF circuits*. Artech House, 2010.
[32] "Small portable aes128/192/256 in C," Github, 2013, https://github.com/kokke/tiny-AES-c/.
[33] P. Juszczak, D. Tax, and R. P. Duin, "Feature scaling in support vector data description," in *Proc. asci*. Citeseer, 2002, pp. 95–102.
[34] M. Brisfors and S. Forsmark, "Deep learning side-channel attacks on AES," Master's thesis, School of Electrical Engineering and Computer Science, KTH, 2019.
[35] G. Perin, B. Ege, and J. van Woudenberg, "Lowering the bar: Deep learning for side-channel analysis (white-paper)," in *Proc. BlackHat*, 2018, pp. 1–15.
[36] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.