# Automated Probe Repositioning for On-Die EM Measurements

Bastian Richter
*Ruhr University Bochum*
*Horst Görtz Institute*
Bochum, Germany
bastian.richter@rub.de

Alexander Wild
*NXP Semiconductors*
Hamburg, Germany
alexander.wild@nxp.com

Amir Moradi
*Ruhr University Bochum*
*Horst Görtz Institute*
Bochum, Germany
amir.moradi@rub.de

***Abstract*—In side-channel analysis attacks, on-die localized EM monitoring enable high bandwidth measurements of only a relevant part of the Integrated Circuit (IC). This can lead to improved attacks compared to cases where only power consumption is measured. Combined with profiled attacks which utilize a training phase to create precise models of the information leakage, the attacks can become even more powerful. In contrast, localized EM measurements can cause difficulties in applying the learned models as the probe should be identically positioned for both the training and the attack even when the setup was used otherwise in between. Even small differences in the probe position can lead to significant differences in the recorded signals.**

**In this paper we present an automated system to precisely and efficiently reposition the probe when performing repeated measurements. Based on the training IC, we train a machine learning system to return the position of the probe for a given measurement. By taking a small number of measurements on the IC under attack, we can then obtain the coordinates of the measurements and map it to correct the coordinate system. As the target for our practical analyses, we use an STM32L0 ARM-M0+ microcontroller with integrated hardware AES.**

***Index Terms*—Side-channel analysis, EM probe, convolutional neural network, machine learning**

## I. Introduction

Shortly after the introduction of the power side-channel also the electro-magnetic emanation (EM) resulting from the current flow was identified as a source of side-channel signals. A main advantage is its high bandwidth which is less influenced by parasitic capacitances introduced by the board or the chip's package. Especially, if measured directly on the chip package or even better directly on the die of a decapped chip, it can reaveal more information than the externally measured power consumption. As introduced in 2001 [1], on-die measurements with very small probes in the range of a few hundred micrometer, can measure a localized signal of a part of the chip. This enables the attacker to measure an isolated signal emitted by the targeted circuit (e.g., an encryption core) not influenced by the noise generated by other parts of the chip (e.g., by the CPU core running in parallel to the encryption).

Another improvement to the initial unprofiled power analysis attacks like Differential Power Analysis (DPA) [2] and Correlation Power Analysis (CPA) [3] are profiled attacks especially template attack [4]. For these attacks an identical chip is required which can be controlled by the attacker. The controllable chip is used to create a leakage-model which is further used for a more precise and thus more efficient attack. In some cases the leakage-model is even able to directly target values and not only their power-model like the Hamming Weight (HW) in CPA. There has been research on profiled attacks extending from the original multivariate Gaussian distribution fitting to other machine learning techniques like Support Vector Machines (SVMs) [5] or deep neural networks.

At first, a profiled attack based on localized EM measurements seems like a good combination, as the local signal should improve the profiling by excluding signal sources not related to the target value. But the downside is that performing the measurement on the attack chip at exactly the same position as on the training chip can be very difficult. As the signal highly varies with the position of the probe, even a slight misplacement can decrease the effectiveness of an attack. To counteract this, the leakage-model need to be made more robust, e.g. by pre-processing the traces. But still the question arises whether the attack could be better optimized if an exact repositioning is possible.

Especially, security evaluation labs which try to determine the physical resistance of a device face several scenarios in their daily business which requires accurate probe repositioning. The previously mentioned attacks can address various target values. To identify the best attack position of a potential target value, typically a grid scan is performed by measuring EM traces from each probe position in a grid which have to be further analysed. In addition, to mount the previously mentioned attacks usually various EM tracesets with different input data patterns are required which are typically not measured in a one-shot. Due to the high computational complexity and hence runtime of the trace analysis respectively attacks, evaluation labs frequently swap devices during the trace analysis respectively attack process to increases the measurement setup utilization. Another scenario is simply the reevaluation of software-based implementations that include fixes of previously detected weaknesses.

In recent years, machine learning algorithms and especially deep learning are gaining more and more attention due to the impressive results in the field of image processing like

This work is partly supported by the German Research Foundation (DFG) through the project 393207943 ”Security for Internet of Things with Low Energy and Low Power Consumption (GreenSec)”, and Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

object detection or image classification which can be seen as pattern detection. Also in the side-channel field deep learning has produced some interesting results [6], [7] like being immune to jitter if used with convolution layers [8] and being able to attack masked implementations in a supervised and unsupervised [9] setting.

### A. Contribution

In this paper we show that it is possible to train a convolutional neural network to recover the probe position of a given EM trace. Based on the neural network prediction it is hence possible to implement a simple algorithm on top, to accurately reposition an EM probe on a target chip. The practical evaluation is performed on a modern microcontroller targeting a software and a hardware implementation of the AES encryption.

## II. Background

### A. Neural Networks

Neural networks transfers its input data from one domain into another domain which represents the target of typical applications like classification and regression. The basic unit of neural networks are neurons which are typically organized in layers. A neuron receives its inputs either from other neurons of the previous layer or from an external source. Typically, every neuron receives the output of every neuron in the previous layer. These layers are thus called fully connected and a network only consisting of these layers is called Multilayer Perceptron (MLP). Other architectures with different connection schemes are also possible, e.g. parallel layers with different properties then connecting to a common successor. In the neurons, inputs are multiplied with associated weight values and summed. By weighting the inputs, a relative importance is given to them.

The sum is further processed by a simple non-linear function, called activation function, which adds non-linearity to the network and hence the capability for a non-linear domain mapping. Each layer of neurons changes the representation of the data and performs a step towards the domain transfer. Based on the target domain, a layer can compress or decompress the data, transform it to a more abstract or detailed representation.

In case of supervised learning, a set of labeled data ,i.e., input data with corresponding output is given to the training process. A loss function is defined which calculates a value related to the error between the value predicted by the network and the correct value. This error is then propagated back through the network to minimize the loss by adjusting the weights. Stochastic gradient descent [10] and its advanced versions like Adam [11] are the most common technique for weight optimization but others are possible as well, e.g. evolutionary algorithms.

Parameters that have to be set before the training process which e.g. define the network architecture, configure the backpropagarion algorithm, or preprocess data are called hyperparameters. Typical architecural hyperparameters are the properties of the layers like the number of neurons and the activation function. Since hyperparameters are fixed at training time but need to be optimized for an application, multiple training interations with different hyperparameters are needed to optimize the efficiency of the neural network.

### B. Convolutional Neural Networks

In unstructured data it can happen that the information required to perform the domain transfer is not always located at the same position. To address this problem, neural networks make use of convolution layers. Those layers define filters ,i.e., a set of neurons that stride along the data and search for this information. Technically, a convolution layer groups its neurons while the weigths are shared between the groups. Usually, a convolution layer expands the data and hence convolution layers are often combined with pooling layers that perform a compression by removing the spacial information of the filter outputs, e.g., by reducing a dimension by keeping only the maximum of a certain interval.

## III. Probe Repositioning

When switching the ICs in a typical side-channel measurement setup, often either the whole PCB is switched or a socket for the IC is used. This introduces some variation in the positioning of the IC relative to the stage and thus the coordinate system in which the probe is moved. The same holds for sockets which also have some tolerance for easier insertion.

### A. Visual Positioning

The most simple and widely used method is visual positioning of the probe using a microscope mounted above. The main downside of this method is its precision. For automated positioning, the camera also needs to be mounted static in relation to the probe. Manual positioning is also often performed by orienting on structures on the IC. But this needs visual clues be present for orientation which might not exist near the point of measurement if a shield is present on high security ICs or if approaching the IC from the backside.

### B. Scan of The Chip

The next method is scanning over the chip and correlating the profiling traces with the traces measured during the scan. This might be combined with a coarse visual prepositioning. In theory, this method should lead to the highest precision as it exhaustively tries to find the position most similar to the profiled one. In practice, jitter or random timing can prevent a correlation to properly work. Additionally, it also takes a long time as scanning over the chip and doing measurements for correlation is slow.

### C. Direct Positioning

Ideally, it would be possible to map the coordinate system of the profiling chip to the attacked chip by only a few measurements. This can be done by taking measurements of at least two points on the attacked chip and then find their position on the profiling chip, by correlating over a scan. Then the profiling coordinate system can be mapped to the one of

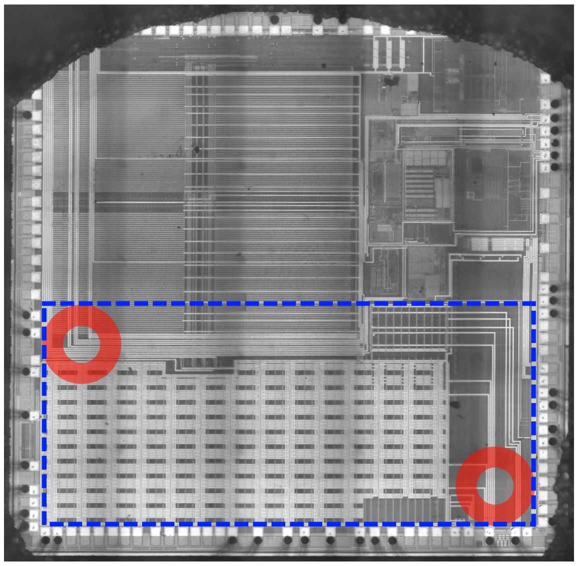


Fig. 1: Stitched microscope image of the decapped STM32L081CB with marked start and end position of the probe (red circles) and scan area (blue dashed rectangle). The red circles approximate the size of the probe's coil.

the target chip. This method is as slow as the second one when repositioning is only needed once, but more efficient if it is done multiple times. To improve it, we can find a function which directly maps a measured trace to a position as this would skip the time-consuming step of correlating over the whole scan. This is the approach we will follow in this work. Based on machine learning we try to find this function by regression to map the trace to coordinates.

## IV. Implementation

### A. Target

Our target platform is a STMicroelectronics STM32L081CB ARM Cortex-M0+ microcontroller [12] placed on a custom measurement board for communication via a USB to UART interface. It features an AES hardware implementation taking 213 clock cycles for one encryption. The LQFP32 package was opened from the front side using nitric acid to expose the die for measurement with the EM probe.

We scanned over the area marked in Figure 1 with a blue, dashed rectangle. Within this area, the logic is covered with the crossed power distribution network which we also suspect to cover the SRAM. The large square block in the top of the image is the flash and the many small structures right of it are analog blocks.

### B. Measurement Setup

For the measurements we used a Langer EMV ICR HH150-27 near-field probe with an inner diameter of $150\,\mu\mathrm{m}$ and a bandwidth of $1.5\,\mathrm{MHz}$ to $6\,\mathrm{GHz}$. The size of the probe in relation to the die is approximated in Figure 1 by the red circles. For automated and precise positioning the probe is

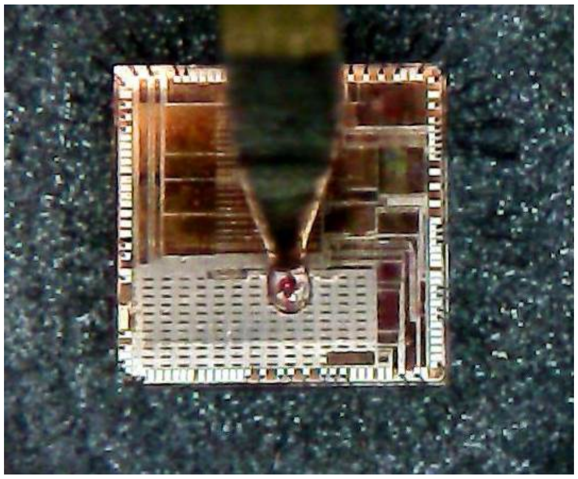


Fig. 2: Photo taken with the measurement setup of the probe on the die of the microcontroller.

mounted on an X-Y-Z stage consisting of Thorlabs MTS50-Z8 axes with bidirectional repeatability of $1.6\,\mu\mathrm{m}$ and backlash of $6\,\mu\mathrm{m}$. Due to mechanical instabilities we expect the total repeatability to be in the range of $20\,\mu\mathrm{m}$. The signal was then recorded using a Teledyne-Lecroy Waverunner 8254M with its full bandwidth of $2.5\,\mathrm{GHz}$ and a sampling frequency of $5\,\mathrm{GHz}$.

### C. Datasets

There are different trace sets needed for the analyses we perform in this paper. For each target, i.e. for the software and for the hardware AES encryption, we recorded the following data sets:

(A) 5000 random positions on a grid of $5\,\mu\mathrm{m}$ in the area marked in Figure 1 with 200 traces recorded for each position for training and validation during training for training.

(B) Scan with a grid of $20\,\mu\mathrm{m}$ (5084 positions) over area marked in Figure 1 with 200 traces recorded for each position for testing of the trained algorithm on training chip.

The Training sets were recorded with random input to also capture a scenario in which the plaintext can not be fully controlled.

### D. Choice of Machine Learning Algorithm

As the positioning system is based on learning a regression function, also other machine learning methods might be applicable. Especially, SVMs and Random Forrests (RFs) are popular methods which have already been used in the side-channel field and also support regression. However, they have the downside of being sensitive to misalignment and jitter. In contrast, Convolutional Neural Networks (CNNs) have been shown to be able to overcome jitter and random timing in side-channel attacks [8]. Thus, we assume that they are also able to perform our regression in the presence of jitter.

EM measurements are especially susceptible to jitter, since the peaks can be very short ($<1\,\mathrm{ns}$) for newer technologies so that even minimal jitter results in peaks not overlapping anymore over multiple traces. We noticed jitter in our measurements,additionally increased by slow IOs of the trigger, which leads to peak positions differing around 10 sample points ($2\,\mathrm{ns}$). Due to these properties, we decided to follow the CNN approach and omit the other techniques.

## E. Neural Network Architecture

Our goal is to find a function to recover the coordinate a given trace was measured at. As we do not want to be constrained to a certain grid on the axes, we decided not use a classification to the coordinates. Instead, we formulated the problem as a regression to two values. These represent the two axes and have a range from 0 to 1 which represents the whole range of the axis.

In our case the lengths of the axes differ which would result in different scale factors. As the loss function weights both axes the same, the shorter axes would have a higher influence on the loss value. To counteract this, we scaled both axes so that 1 represents the maximum of the longest axis which results in Equations 1 and 2 for the coordiante labels $l_x$ and $l_y$ if the x-axis is longer.

$$l_x = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1) \qquad l_y = \frac{y - y_{min}}{x_{max} - x_{min}} \quad (2)$$

Similarly, the traces are also scaled to a range of 0 to 1 with 0 (1, respectively) representing the minimum (maximum, respectively) of the ADC values. As argued in the previous section, we used a CNN for the regression. In addition to the previously mentioned advantages, CNNs lower the complexity for inputs with a high dimensionality, as they share their weights and thus have fever parameters to train. This is favorable for us as our inputs consist of multiple 1000s points.

Figure 3 visualizes the CNN architecture used which consists of two convolution blocks (1-3 and 4-6) and two dense layers. The input is first normalized by a Batch Normalization layer before passed to the first convolution block. This improves the training as the range of the different points highly differs. Points representing peaks in the trace have a high mean and variance while the others mean and variance is much lower. The two convolution blocks consist of a 1D convolution layer (1 and 4) followed by a 1D maximum pooling layer (2 and 5). To counteract overfitting we also added a dropout layer(3 and 6) to the blocks. The two blocks are followed by a dense layer (7). All previous layers have ReLu as activation function, only the last dense layer (8) uses a Sigmoid activation to constrain the output of the network to the range $[0, 1]$. The output of the Sigmoid layer (8) is then used with the mean-squared-error loss function. We implemented the network in Python using Keras [13] with TensorFlow [14] backend.

This general CNN architecture was used for both targets but we performed independent hyperparameter optimizations for which we used the Talos framework [15]. Table I lists the final parameters of the layers used for the evaluations.

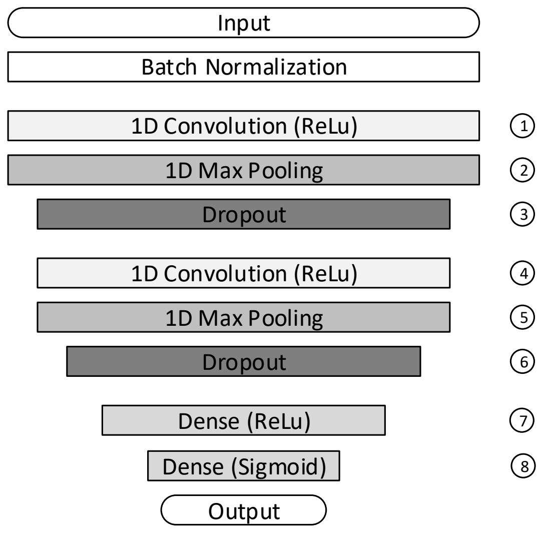


Fig. 3: Convolutional Neural Network architecture used for position recovery.

| No. | Layer | Software AES | Hardware AES |
|---|---|---|---|
| 1 | 1D Convolution | (30, 50) | (20, 10) |
| 2 | 1D Max Pooling | 5 | 2 |
| 3 | Dropout | 0.2 | 0.2 |
| 4 | 1D Convolution | (50, 20) | (30, 30) |
| 5 | 1D Max Pooling | 5 | 2 |
| 6 | Dropout | 0.2 | 0.2 |
| 7 | Dense | 64 | 64 |
| 8 | Dense | 2 | 2 |

TABLE I: Hyperparamter for software and hardware AES CNN Architecture. The format for Convolution layers is (no. filters, kernel size).

## F. Points of Interest

As the traces for our targets are very long with 70,000 (HW) and 150,000 (SW) points, we selected two areas of interest from them which are marked in Figures 4 and 6. These were picked after a quick visual inspection. The areas in the software traces were picked to contain parts of each sub operation of the AES round, i.e. SubBytes, ShiftRows, and MixColumns. For the hardware AES we picked the beginning and end of the trace, as these also include parts of the control code for the hardware which we suspected to give additional position dependent information. Another possible approach would be to pick parts which exhibit a high variance over different position on the chip, but as our coarse selection already worked well we did not evaluate this approach.

## V. Practical Results

## A. Software AES Encryption

Our first target is a timing constant software implementation of the AES encryption written in ARM Thumb assembly. We measured traces of the first round of the AES which are shown in Figure 4. The different parts of the round are clearly

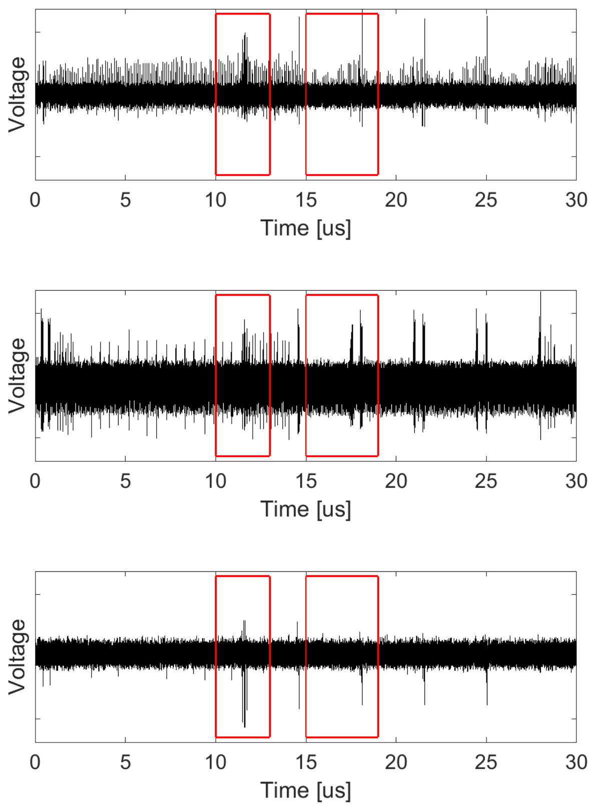


Fig. 4: Example traces of the software AES (first round) for different positions within the scan area. Red lines mark the parts of the traces used for training and recovery.

discernible within the trace. Also, traces of different positions highly differ and peaks are only present for certain operations.

We trained the CNN on dataset (A) which contains random positions. The traces (150,000 points) were preprocessed by cutting them to the marked areas (35,000 points in total) of interest and calculating the mean of 20 traces each, so we get 10 mean traces for each position we use for training.

To establish a baseline of the accuracy of the position recovery, we used dataset (B) which contains measurements on a grid of $20\,\mu\mathrm{m}$ from the training chip. After preprocessing, we fed the traces into the trained net and calculated the mean Euclidean distance (over the 10 mean traces per position) to the point they were measured at. The result is plotted as a heat map in Figure 5. There are some small spots with an increased distance of around $250\,\mu\mathrm{m}$ but the over all mean is $43.86\,\mu\mathrm{m}$ with a standard deviation of $23.58\,\mu\mathrm{m}$. The histogram also poofs that there are only few positions with a high distance as the center of the highest bin is $30\,\mu\mathrm{m}$ with a bin range of $5\,\mu\mathrm{m}$.

The distance is in general higher on the right side of the map wich corresponds to the layout of the chip. In Figure 1 the power grid of the logic area does not continue to the right end of the scan area but there is a different structure and some power lines coming from the VDD and GND bond pads. As we suspect that the power lines are the main source of electro

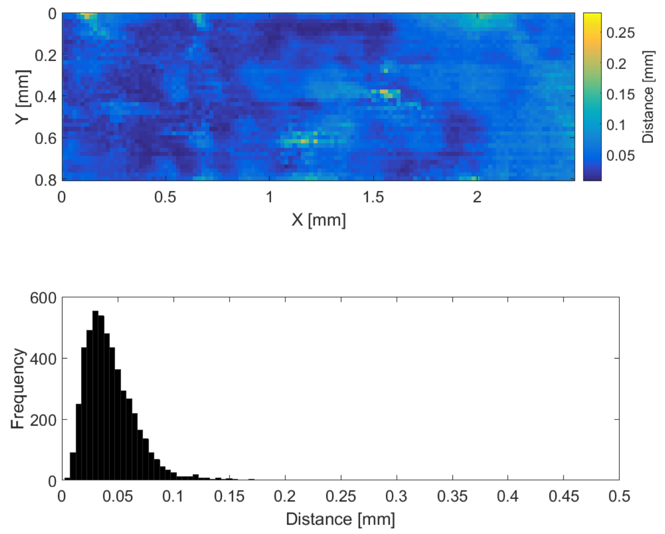


Fig. 5: Mean distance of recovered position over the scanned area on the training chip (top) and its distribution (bottom) for software AES.

magnetic radiation in this area, we expect that there is less operation and hence location depended leakage at this part because the power lines carry the total current of a larger area. Consequently, the mean distance is only $38.08\,\mu\mathrm{m}$ with a standard deviation of $19.31\,\mu\mathrm{m}$ if only the logic area is considered.

Please note that the scan we show do not directly correspond to the area marked in Figure 1 which is defined by the outer edge of the probe at their maximum positions. As the probe is considerably larger (inner diameter of $150\,\mu\mathrm{m}$) than our step size of $20\,\mu\mathrm{m}$ which corresponds to the pixels in the map (Fig. 5), we can not directly map it to the area in the photo but have to consider it as a map over the range of movement.

## B. *Hardware AES Encryption*

The STM32L081CB also features a hardware implementation of AES which takes 213 cycles to perform an encryption. The key and plaintext are loaded in software by shifting these into the AES register in 32-bit words. This writing to the data registers and the later reading of the ciphertext is included in the traces shown in Figure 6. The full traces are 70,000 points long and we picked 33,000 points for training which include the writing and reading of the data registers. As the AES core is expected to be smaller and thus less far distributed than the ARM core, we expect signals with a lower amplitude which is confirmed by the traces in Figure 6.

Following the same approach as with the software implementation, we evaluated the network on how good it can recover the position on the training chip. The map in Figure 7 behaves similarly to the software implementation but with the high distance outlier only on the right edge of the map. The impact of the different chip structure on the right is thus much higher with a small mean error of $25.16\,\mu\mathrm{m}$ with a standard

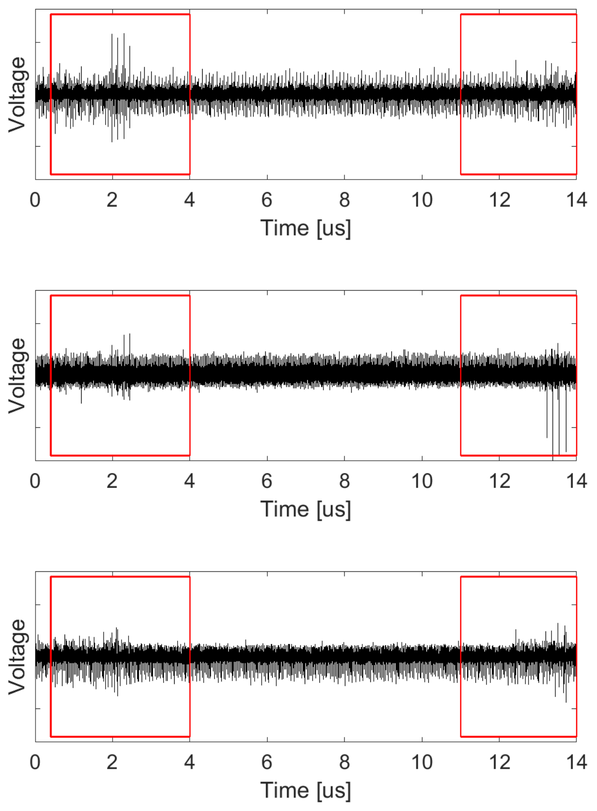


Fig. 6: Example traces of the hardware AES (whole encryption) for different positions within the scan area. Red lines mark the parts of the traces used for training and recovery.

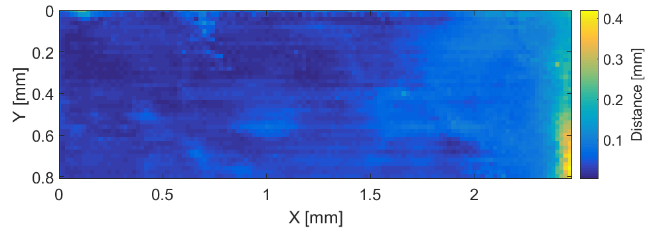
deviation of 17.82 µm over the logic area and an increasing gradient on the right side. The overall mean of the error is 50.90 µm with a standard deviation of 43.50 µm. This is also confirmed by the histogram with its highest bin's center also at 30 µm but with a second small peak at 70 µm originating from the increased distance on the right.

## VI. Conclusion

Starting with a scan over the chip which is usually already performed when starting an EM analysis of a chip we have shown that by training a convolutional neural network it is possible to recover the position at which a given trace was measured. The accuracy is in the range of less than 50 µm with a distance of around 30 µm for the most positions. This accuracy has been achieved for a software as well as a hardware implementation of AES on a modern microcontroller platform. Therefore, the system enables repositioning to repeat measurements of a chip.

### A. Future Works

An interesting future work would be to see whether the system is still possible to recover the position for a target chip which features strong countermeasures, as these often feature high temporal misalignment which makes it much more difficult to find features an extract their information. For side-channel leakage it has already been shown but it might be different for this application.



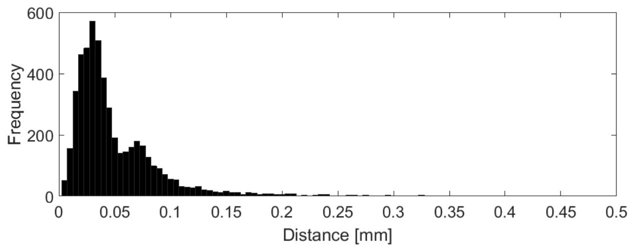


Fig. 7: Mean distance of recovered position over the scanned area on the training chip (top) and its distribution (bottom) for hardware AES.

## References

[1] Karine Gandolfi, Christophe Mourtel, et al. Electromagnetic analysis: Concrete results. In *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.
[2] Paul C. Kocher, Joshua Jaffe, et al. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
[3] Eric Brier, Christophe Clavier, et al. Correlation Power Analysis with a Leakage Model. In *CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
[4] Suresh Chari, Josyula R. Rao, et al. Template attacks. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
[5] Gabriel Hospodar, Benedikt Gierlichs, et al. Machine learning in side-channel analysis: a first study. *J. Cryptographic Engineering*, 1(4):293–302, 2011.
[6] Zdenek Martinasek, Jan Hajny, et al. Optimization of power analysis using neural network. In *CARDIS*, volume 8419 of *Lecture Notes in Computer Science*, pages 94–107. Springer, 2013.
[7] Houssem Maghrebi, Thibault Portigliatti, et al. Breaking cryptographic implementations using deep learning techniques. In *SPACE*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016.
[8] Eleonora Cagli, Cécile Dumas, et al. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.
[9] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):107–131, 2019.
[10] David E Rumelhart, Geoffrey E Hinton, et al. Learning representations by back-propagating errors. *Nature*, 1986.
[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
[12] STMicroelectronics. *STM32L081CB Datasheet*, 2017. Rev 5.
[13] François Chollet et al. Keras. https://keras.io, 2015.
[14] Martín Abadi, Ashish Agarwal, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
[15] Autonomio. Talos. https://github.com/autonomio/talos, 2019.