Automatic Extraction of Secrets from the Transistor Jungle using Laser-Assisted Side-Channel Attacks

Conference Paper · August 2021

CITATIONS
2

READS
446

4 authors, including:

Shahin Tajik
Worcester Polytechnic Institute
57 PUBLICATIONS **694** CITATIONS
SEE PROFILE

Jean-Pierre Seifert
Technische Universität Berlin
159 PUBLICATIONS **3,033** CITATIONS
SEE PROFILE

Automatic Extraction of Secrets from the Transistor Jungle using Laser-Assisted Side-Channel Attacks

Thilo Krachenfels*, Tuba Kiyan*, Shahin Tajik† and Jean-Pierre Seifert*‡
* *Technische Universität Berlin, Chair of Security in Telecommunications*
† *Worcester Polytechnic Institute, Department of Electrical and Computer Engineering*
‡ *Fraunhofer SIT*

Abstract

The security of modern electronic devices relies on secret keys stored on secure hardware modules as the root-of-trust (RoT). Extracting those keys would break the security of the entire system. As shown before, sophisticated side-channel analysis (SCA) attacks, using chip failure analysis (FA) techniques, can extract data from on-chip memory cells. However, since the chip's layout is unknown to the adversary in practice, secret key localization and reverse engineering are onerous tasks. Consequently, hardware vendors commonly believe that the ever-growing physical complexity of the integrated circuit (IC) designs can be a natural barrier against potential adversaries. In this work, we present a novel approach that can extract the secret key without any knowledge of the IC's layout, and independent from the employed memory technology as key storage. We automate the – traditionally very labor-intensive – reverse engineering and data extraction process. To that end, we demonstrate that black-box measurements captured using laser-assisted SCA techniques from a training device with known key can be used to profile the device for a later key prediction on other victim devices with unknown keys. To showcase the potential of our approach, we target keys on three different hardware platforms, which are utilized as RoT in different products.

1 Introduction

For security applications, people rely on hardened hardware modules, like Trusted Platform Modules (TPMs), as the root-of-trust (RoT) for storing secret keys. Those keys ensure the functioning of complex and delicate systems like routers, servers, sensor systems, and cars by establishing secure communication channels, safeguarding trusted code execution, and protecting the intellectual property embodied in the device. Extracting secret keys managed by a RoT hardware would break the entire system's security. Possible motivations for attackers are the extraction of secret information, tampering with the design, or cloning the device.

Modern integrated circuits (ICs) and system-on-chips (SoCs) consist of billions of transistors, which makes the reverse engineering of the design and layout very challenging. Moreover, data extraction from various key storage technologies requires different measurement tools and expertise, making the attack costly and unscalable. This physical complexity might lead to a belief by vendors that the localization and extraction of assets/secrets on their products is a laborious task. In addition to that, the usage of the keys in diverse applications, such as firmware/bitstream decryption, asymmetric cryptographic operations, or logic deobfuscation, makes the generalization of an attack against RoTs infeasible.

There are companies like Techinsights [1] and Texplained [2] that invest lots of expertise and effort into fully reverse engineering ICs with destructive techniques and using sophisticated failure analysis (FA) tools, such as scanning electron microscopes (SEMs) and focused ion beams (FIBs) [3]. They can extract the IC's netlist, analyze its functioning, and therefore, find the location where the key is stored. While effective, this approach is very time consuming and expensive. On the other hand, researchers have shown that attacks on some specific devices only require partial reverse engineering. Applying SEM [4, 5], FIB [6], microprobing [7], and laser-assisted side-channel analysis (SCA) techniques using laser scanning microscopes (LSMs) [8, 9, 10, 11] are examples of such academic work. Nevertheless, these attacks have only been carried out in an experimental environment, where many details of the design were available beforehand or had to be gathered manually.

Considering the high amount of manual reverse engineering work, one might ask if machine learning techniques could be applied in the context of hardware security to reduce the required knowledge for key extraction. Indeed, the benefit of applying deep learning techniques on classical SCA attacks, like power and electromagnetic (EM) analysis, have already been discovered and studied extensively [12, 13, 14, 15]. At the same time, convolutional neural networks (CNNs) have become the default choice for image classification tasks, as they remove the need for manually tailoring the algorithm to

its specific application. Consequently, CNNs could also be one suitable method for extracting a key from images captured by FA techniques from a complex chip with unknown design and layout. In other words, if an attacker combines image recognition techniques with sophisticated FA tools that are capable of capturing the logic state from inside the IC, a new threat dimension arises. Such an approach can antiquate the expensive reverse engineering portion of hardware attacks. On the positive side, such a tool, if automated, can also be used for security assessment of products.

Our contribution.

In this work, we develop an attack approach drawing the connection between image recognition techniques, profiling SCA, and sophisticated FA tools to extract the secret key from memory cells of an IC without requiring any knowledge about the chip's layout and its functioning.

To validate our claims, we conduct SCA using two different and well-known laser-assisted SCA methods, namely thermal laser stimulation (TLS) [10] and laser logic state imaging (LLSI) [11]. We apply these SCA techniques on three different hardware targets with various process technology sizes: the dedicated key memory of an 20 nm Field Programmable Gate Array (FPGA), the SRAM of a 180 nm microcontroller, and the registers of an 60 nm FPGA. All these platforms can be potentially part of an RoT implementation. To showcase the strength of our approach, we exemplarily deploy CNNs to create models out of obtained measurements from these devices. The results demonstrate that our trained models can extract an unknown secret key from the victim devices with high accuracy, even in the presence of largely irrelevant information and activities on the chip. Moreover, it is not required to know the location of targeted memory cells and how to interpret the bit values from the measurements. Note that our approach is not limited to optical SCA attacks, and can also be combined with SEM, FIB, or any other FA microscopy tools, which capture the activity of transistors.

While in this work we have applied deep learning due to its straight-forward nature for highlighting the threat of our approach, deploying other image recognition techniques is also conceivable. In this regard, we are open-sourcing the side-channel data to enable other researchers to improve data extraction using various techniques. Consequently, we would like to stress that the emphasis of this work is on showing that laser-based SCA can eliminate the reverse engineering step for extracting secret information, and not on applying deep learning techniques as profiling SCA tool.

2 Threat Model

2.1 Target

In hardware RoT applications, we can distinguish between two different kinds of keys. At least one *root key* must be stored in plaintext in a non-volatile memory (NVM). Other keys might be stored internally/externally in an encrypted form, decryptable by the root key. In the following, we will refer to them as *application keys*. In addition to its usage as key-decryption-key, the root key might also be deployed directly, e.g., for firmware or bitstream decryption. Since the root key is typically stored in a (tamper- and read-proof) NVM, such as flash memory, EEPROM, or ferroelectric RAM, the direct extraction of its content is not a straightforward task [5]. However, even if the NVM is considered secure, for being used, the contained key will be loaded into CMOS memory cells at some point in time, see Fig. 1. The same holds true for application keys after they have been decrypted.

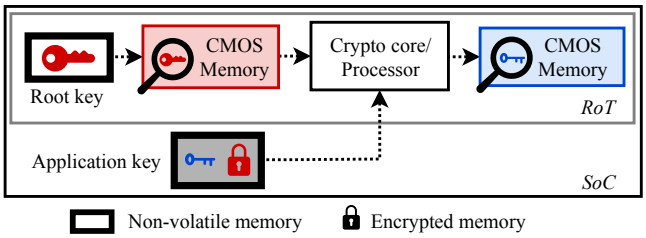
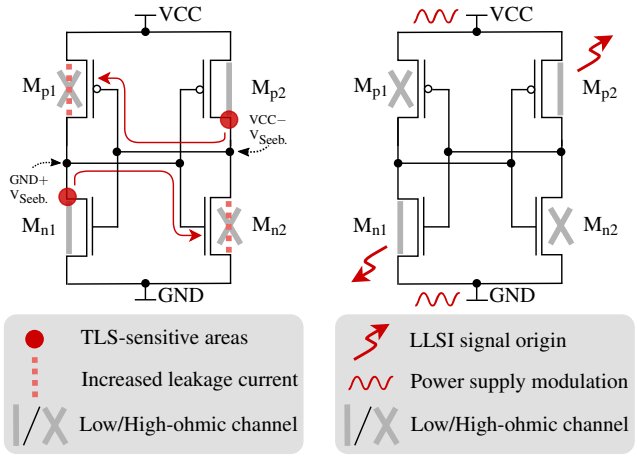


Figure 1: Extraction of the root key after it was loaded from a tamper- and read-proof non-volatile memory, or of an application key after it has been decrypted using the root key.

Previous work has shown that sophisticated non- and semi-invasive FA tools are capable of extracting logic states from CMOS logic gates [16] and memory cells [8, 10, 11]. These techniques typically produce an image (i.e., activity map or response image) which contains information about the logic state of the area of interest. Yet, extracting the actual memory content from these images can be a challenging task, even if the chip's layout is known, or at least understood to a certain degree. Although tools like SAT solvers [11] and image recognition techniques can aid the localization of the key, much prior knowledge of the memory cell's design, its geometry, and its exact location is required. Therefore, a potential attacker might be highly motivated to reduce the effort for extracting keys from the images.

2.2 Attacker's Motivation

We assume an adversary who has access to FA tools and has a strong motivation to avoid expensive reverse engineering of the whole IC for just extracting a single key out of it. One might ask why an adversary would invest that much effort into extracting the key from a single device. The primary motivation in many scenarios is that the same key is used for all devices, for instance, when firmware, bitstream, or logic encryption is used to protect the proprietary design of a system. The key is therefore programmed by the vendor before the product is shipped to the customer. Consequently, extracting the key from one device would break the security of all devices from the same family. Even if the key differs between



(a) TLS (b) LLSI

Figure 2: Schematic of a CMOS memory cell and how the two measurement techniques can extract the cell's logic state. Transistors for read and write access are omitted. Figures based on [19, 20].

devices, it should be kept in mind that all chips from a device family have the same layout. Therefore, the adversary can learn how to extract the key from a training device and use her knowledge to extract the key also from other devices of the same family.

3 Background

3.1 Optical Side-Channel Analysis Attacks

For being able to debug the active silicon of integrated circuits (ICs) in the presence of the many metal layers on the chip frontside, techniques have been developed to access on-chip signals through the IC backside [17]. The corresponding optical side-channel analysis (SCA) techniques take advantage of the high infrared transmission in silicon for wavelengths above 1 µm, basically allowing to "see through" the bulk silicon at the IC backside. Due to their availability in FA labs around the globe, related techniques like photon emission analysis, laser stimulation, and optical probing have been adopted by the hardware security field [18, 10, 9]. A typical setup consists of a laser scanning microscope (LSM) with laser sources of different wavelengths, a detector for measuring the reflected laser light, and optionally a camera for photon emission analysis.

The two relevant techniques for this work, including reported attacks in the literature, will be discussed below.

3.1.1 Thermal Laser Stimulation

Thermal laser stimulation (TLS) is an SCA technique that induces electrical perturbations on a target device by creating local temperature gradients when stimulating an area of interest with a laser beam. The laser beam's wavelength is above 1.1 µm, which does not have enough energy to generate electron-hole pairs, but thermal gradients. A temperature variation on a thermocouple can lead to a voltage generation, which is known as the Seebeck effect [21]. The Seebeck voltage can be leveraged to extract the logical states from CMOS memory cells [19].

A CMOS memory cell consists of two cross-coupled inverters, with one transistor per inverter being low-ohmic (conducting) and one being high-ohmic (nonconducting), see Fig. 2a. Hence, while storing a value, i.e., in the stable state, only a negligible current is flowing between VCC and GND. However, if a laser beam stimulates the drain-bulk junction of a transistor with low-ohmic channel, it generates a Seebeck voltage ($V_{Seeb.}$). This voltage is forwarded along the circuit to the gate of a transistor in the high-ohmic state. This transistor is slightly switched on and – via exponential sub-threshold operation–, the current drawn from the memory cell's power supply increases. If an area of interest on the device is scanned pixel-wise by a laser beam and the small power consumption variations are recorded along with the laser beam's location, the TLS response map of the scanned area can be obtained. The areas of the two sensitive transistors will show up brighter in the TLS response map, due to the slight increase in power consumption. For the opposite bit state, the other two transistors will appear on the response map, making the two different bit states of the memory cell distinguishable from each other.

TLS is a well-understood technique that has been used to read out SRAM memory on microcontrollers [19, 22] and extract the cryptographic key from the battery-backed RAM on an FPGA [10]. One scan over the area of interest can reveal the entire memory content, and therefore, TLS can be considered a single-trace SCA technique. Naturally, the memory content should stay constant during the scan. Recently it has been shown that TLS can be mounted with cheaper setups than previously expected – for around $100k [23].

3.1.2 Laser Logic State Imaging

Optical probing is an FA tool used for acquiring electrical information from inside the IC [24, 8, 9]. Electro-optical frequency mapping (EOFM) is an optical probing technique that allows creating a 2-D activity map of circuits, showing nodes that are switching at a particular frequency [25]. While light with wavelengths above 1 µm scans the IC backside pixel by pixel, it passes through the silicon substrate. The light is partially absorbed and partially reflected by structures such as metal layers and transistors, whereas the electrical field present at transistors influences the light's amplitude and phase. A portion of the reflected light leaves the IC through the backside where it is converted into a voltage and fed into a narrow-band frequency filter set to the frequency of interest. The resulting signal's amplitude and the position information

form the 2-D activity map on which areas modulating at the frequency of interest appear as bright spots.

For EOFM measurements, it is necessary to know the internal switching frequency of the circuit of interest to track the signals. This frequency can be hard to predict, and even worse, there is not necessarily any switching activity for memory cells if no read/write operation is carried out. This problem can be tackled by inducing a frequency, for example, by modulating the core voltage that supplies the circuit under test. The corresponding technique is called laser logic state imaging (LLSI) and has been introduced as an extension to EOFM [20]. LLSI makes the extraction of static logic states possible, e.g., from a CMOS memory cell, as illustrated in Fig. 2b. The low-ohmic transistors' electric fields oscillate with the power supply's modulation frequency, and hence, produce an EOFM signal. In contrast, off-state transistors do not produce a strong EOFM signal. Consequently, the logic state of the SRAM cell can be deduced. LLSI has been used to read out SRAM on a microcontroller [22] and the registers on an FPGA [11].

Note that LLSI can be used to extract not only the state of SRAM cells or registers, but also any cluster of transistors, such as buffers or logic gates. As long as the bit state of the logical element affects the involved transistors, the bit value can be extracted. Next to TLS, also LLSI can be considered a single-trace SCA technique, as one scan over the region of interest is sufficient to capture its entire logic state. Similarly, to perform LLSI, the memory content has to remain constant during the scan. One way to achieve this requirement is to halt the clock signal to prevent any update in the values of the memory [11]. However, in some applications, e.g., logic locking, the secret key has to be provided constantly to the locked circuit in order to keep it unlocked during runtime, and therefore, no clock control is needed. Moreover, it has been observed that some cryptographic accelerators do not necessarily clear key registers after encryption/decryption [26], and hence, the key remains in the registers as long as the device is powered on.

3.2 Deep Learning for Image Classification

Due to their high flexibility, convolutional neural networks (CNNs) [27, 28, 29] are a popular choice for many computer vision applications such as image recognition [30, 31]. Image recognition typically consists of two tasks: object classification (also called image-level annotation) and object detection (object-level annotation). While for classification only the presence of an object from a given set of classes is assessed – and not its position –, object detection is typically a more challenging task. In this work, we are only interested in the existence of a logic `0` or `1` in an image, and therefore, we will only cover object classification in the following.

CNNs are a subclass of deep neural networks and complement the fully-connected (FC) networks (also known as multilayer perceptrons) with trainable feature extractors, the so-called *convolutional layers*. A convolutional layer finds features in the image (e.g., corners, edges, etc.) using trainable filters that cover a certain receptive field. The resulting *feature maps* can be fed into subsequent convolutional layers to detect larger features. Intermediate subsampling steps – *pooling layers* – reduce the resolution of the feature maps to decrease the sensitivity to shifts and other distortions. Finally, after some repetitions of convolutional and pooling layers, the output is *flattened* and fed into the FC network to classify the images.

In the literature, different architectural designs for CNNs have been reported, e.g., LeNet-5 [28], AlexNet [32], and VGG [33]. The authors of the VGG architecture presented a generic design consisting of the repetitive application of filters with a very small receptive field (3×3 pixels), followed by a max-pooling over a 2×2 pixel window. The stack of convolutional layers is followed by FC layers with one neuron for each class in the output layer [33]. The structure of multiple small convolutional layers followed by a max-pooling layer is often referred to as *VGG-block* and has become a popular building block and starting point when designing a new model from scratch, like it will be required for the optical key extraction. Different concepts have been developed to reduce over-specialization on the training data (so-called overfitting) of CNNs, especially when only a small training dataset is available. For instance, a *dropout layer* can remove random nodes from the FC layers during training, which leads to the extraction of more robust features [34]. Furthermore, *data augmentation* can increase the number of training samples artificially, and therefore, reduce overfitting as well [35].

3.3 Related Work

This work builds on an approach that is known as profiled side-channel analysis [36], where a device under the adversary's control is used to create a leakage model, which is later used to extract the secret from a similar device [37]. In the literature, profiled SCA is typically applied to a cryptographic core by observing its operation, for instance, through power and EM side-channels. In the *profiling phase*, the behavior of the DUT is observed and incorporated in a leakage model using either statistical methods (a.k.a template attacks [38]) or machine learning techniques [39], such as support vector machines [40] and neural networks [12, 13, 14, 15]. In the *attack phase*, the extracted model is used to extract the unknown secret from the target device. Traditional SCA has limited applicability in some cases, e.g., when the key is not involved in active computations, or when countermeasures prohibit the capturing of a sufficient number of traces.

Next to side-channel analysis, machine learning is also used in many other applications in the field of hardware security [39], for instance, for hardware trojan detection [41] and reverse engineering [3, 42].

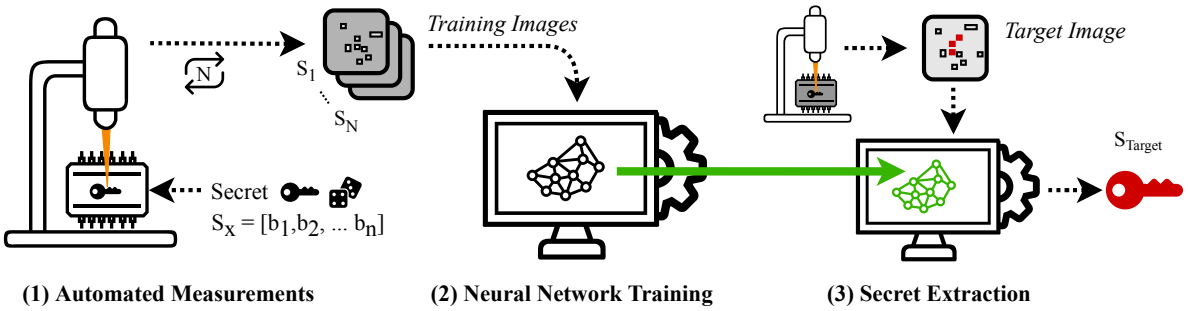


Figure 3: Schematic of the proposed three-step attack approach.

4 Attack Approach

Our attack approach has already been sketched in [43] and assumes that the adversary has access to a training device, for which she can control the contained secret at her will. However, she does not have any knowledge about the design of the chip and the location of the key storage. In this scenario, the approach for the attacker consists of three steps, see Fig. 3. In the first step, randomly chosen keys are programmed into the training device, and SCA images are captured from the IC backside for each key. Subsequently, neural networks are trained with the obtained images. These two steps can be specified as profiling phase. In the final step, the attack phase, the secret on the target device is revealed by one or a few measurements and the previously trained networks. Note that in this work, we chose to apply deep learning techniques for image recognition due to their ad-hoc adaptability to many problems with minimal tuning effort. For the secret extraction from the images, potentially also other machine learning or statistical methods can be applied. In the following, we discuss the three steps of our approach in more detail.

4.1 Automated Measurements

For gathering a training dataset, the adversary captures response images using TLS or LLSI from the training device containing different randomly chosen keys. Since capturing many high-resolution images from larger areas of the chip can be very time consuming, the attacker would first try to find candidate areas for the on-chip memory. Due to the repetitive and regular structure of memory arrays, such candidate areas often can be discovered by analyzing an optical image of the chip. If this is not the case, two response images (containing two different secrets) can be captured from the entire chip area. When subtracting the two images, the attacker can consider all areas showing a difference as candidate areas which should be covered by the automated measurements. Consequently, one sample in the training database consists of one or more response images and the programmed secret. After capturing some samples, the attacker can continue with step 2, that is, training CNNs with the database.

4.2 Neural Network Training

Before training CNNs with the response images, possible drift caused by mechanical instabilities of the setup should be corrected. For this, classical image registration techniques can be used, e.g., by calculating the offset between an optical image captured along with the response image and one fixed optical image. Subsequently, the response image can be transformed according to the calculated shift.

Furthermore, the programmed secret is split into its individual binary bits, which are assigned as multiple labels to each image – one label per bit. Once these preparatory steps are done, a CNN can be designed to learn the secret bits from the response images. More specifically, for each bit of the secret, the images are classified to contain either the binary bit value `0` or `1`. Note that each bit of the secret is handled independently from the other bits. To find out if the images depend on the secret at all, different network architectures should be investigated while trying to learn just a single bit of the secret. Following common practice, we propose to start with a simple model, containing only a few convolutional layers (one VGG-block, see Section 3.2).

To reduce the resources needed for training the model, the images can be split into smaller-sized sections, and a separate model can be trained on each section. As a side-effect, the attacker can find the secret's rough location. If the network does not reach a very high validation accuracy, but the secret bits can be learned to some degree, more measurements from the respective section might be required (supposedly also with higher resolution). The application of data augmentation techniques is likely to reduce the required number of measurements. Once single bits can be learned successfully, a multi-label classification can be attempted to reduce the training time. In other words, one network should learn more than one bit at the same time. This can be achieved by adding more output neurons to the FC network – one per bit of the key to be learned.

4.3 Secret Extraction

When all bits could be learned using the training dataset with a sufficiently high accuracy, the attacker knows the required

locations on the chip and measurement parameters for a successful extraction of the secret. She then can capture response images from the target device (containing an unknown secret) and let the obtained models predict the key from those images. Depending on the accuracy of the network, multiple images with slightly different parameters (like focus position) could be obtained for being able to apply a majority voting scheme on the predicted secret bits, and therefore, achieve a higher probability for predicting all bits of the secret correctly. In this work, we abstain from extracting the secret from a target device and instead rely on the test accuracy from the training phase as an indicator for the attack's success. However, we expect the inter-device differences to be lower than the noise introduced during different measurement runs and by data augmentation.

5 Experimental Setup and Target Devices

In the first part of this section we give details on our setup for conducting TLS and LLSI measurements. Then we briefly describe our setup for the learning part. Finally, we introduce the devices under test (DUTs) and present images of their memory structures captured with our setup.

5.1 Measurement Setup

5.1.1 Optical and Electrical Setup

The core of our setup is a Hamamatsu PHEMOS-1000 FA microscope. It is equipped with a 1.3 µm high-power incoherent light source (HIL) for optical probing and a 1.3 µm laser for thermal stimulation. In addition to the 5×, 20×, and 50× lenses, a scanner-zoom of 2×, 4×, and 8× is available. The light beam is scanned pixel-wise over the device using galvanometric mirrors. For acquiring optical images and conducting LLSI, the reflected light is separated by semi-transparent mirrors and fed into a detector. For LLSI, the detector's output is fed into a bandpass filter set to the frequency of interest. The PC software then produces a 2-D image containing the measured amplitude at each pixel. For conducting TLS measurements, the laser is scanned over the device, and its power consumption is measured using an external current preamplifier (Stanford Research Systems SR 570). The amplifier's output is fed into the PHEMOS PC software, which produces a response map of the locations sensitive to the thermal stimulation. The setup specific to the devices under test is described in Section 5.3.

5.1.2 Measurement Automation

For repeating the measurements with different secrets programmed into the target devices, we programmed a tool in the LabView programming environment. It can control the PHEMOS software (e.g., start and stop measurements, execute auto-focus, move the lens) and access the captured images for correcting horizontal and vertical drift. Furthermore, the tool can trigger the programming of a new secret into the DUT by communicating with a target-specific script running on another PC. In one iteration of the automated measurements, first a new secret is programmed. Then, after executing the auto-focus, an optical image is captured and saved. The drift between that image and the first image of the measurement series is calculated and the lens is moved accordingly. Finally, the TLS or LLSI measurement is conducted and the resulting image is saved along with the secret.

5.2 Learning Setup

For correcting drift in the final images, we made use of the MATLAB image processing toolbox. As machine learning toolbox, we used the Keras API for TensorFlow (version 2.3.0). We ran all our experiments on an Ubuntu 20.04.1 LTS machine with an Intel i7-6850K CPU @ 3.6 GHz, 128 GiB of system memory and a GeForce GTX 1080 Ti GPU with 11 GiB of memory. For all experiments, we made use of the TensorFlow GPU support.

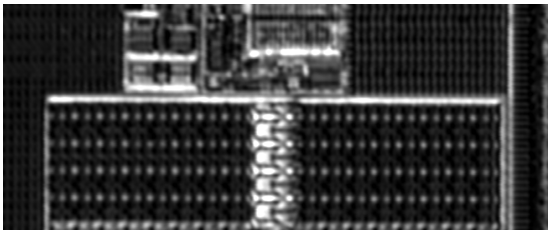
5.3 Devices under Test

We chose three different targets manufactured in different technology sizes and containing different kinds of volatile key memories for our evaluations.

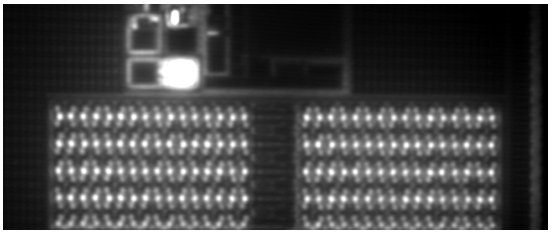
5.3.1 Xilinx Kintex Ultrascale BBRAM

As first and simple target we chose the battery-backed RAM (BBRAM) of a Xilinx Kintex Ultrascale FPGA, which is used for storing a 256-bit bitstream decrpytion key. In principle, BBRAM is identical to common SRAM – except that it is designed to be powered via battery over a long period. Therefore, BBRAM cells are susceptible to optical SCA attacks. In the literature it has been shown that the key from this device family can be extracted using TLS [10].

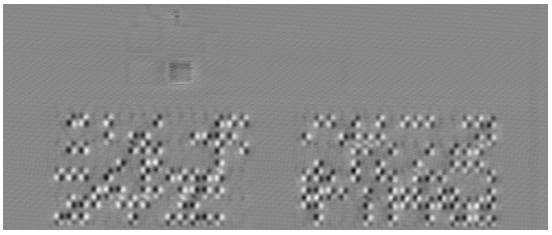
The FPGA, which is manufactured in a 20 nm technology, is mounted on an AVNET development board (AES-KU040-DB-G). The flip-chip package of the FPGA allows direct access to the silicon backside of the chip. For conducting TLS measurements, the current preamplifier is connected to the battery rails of the chip and the main power supply is switched off. The bias voltage of the amplifier supplies the BBRAM during the TLS measurement. For programming a new key, the FPGA has to be powered by the development board's power supplies. To fully automate the programming and measurement process, we made use of the supplies' PMBus interface, allowing to switch the power on and off programmatically via a microcontroller (using the TI PMBus library [44]). Consequently, for programming a new key, the power supplies are



(a) Optical image



(b) TLS response image with a random key programmed



(c) Difference between two TLS response images with different keys

Figure 4: Images of the Xilinx Ultrascale BBRAM.

switched on, a key is programmed via JTAG and the Xilinx Vivado TCL interface [45], and the power supplies are switched off again. Note that during the whole process, the BBRAM voltage is supplied by the current preamplifier. Fig. 4 showcases images of the BBRAM area captured with our setup. Although the chip is manufactured in a 20 nm technology, the size of one memory cell is around 2.8 µm × 3.1 µm, which can be explained by leakage current considerations [10].

5.3.2 Texas Instruments MSP430 SRAM

As second and more flexible target, we chose the freely programmable 1024-byte SRAM of a Texas Instruments MSP430 microcontroller. The chip is manufactured in a 180 nm technology with an SRAM cell size of approximately 2.5 µm × 1.9 µm [22]. The literature shows that the SRAM content of this device can be extracted using TLS and LLSI [22]. For our experiments, we chose to conduct LLSI measurements, as TLS is only possible while the device is in a low-power mode, which is not the case for LLSI. Hence, LLSI can be considered a more powerful technique in this case.

To access the chip backside, the device had to be opened and soldered backside-up on a custom PCB. Note that polishing or thinning the silicon backside was not necessary. For modulating the power supply of the SRAM memory, we made use of the VCORE pin, which provides access to the internally generated core voltage of the microcontroller. To this pin, we connected our modulator circuit, consisting of a voltage regulator whose feedback path is modulated using a laboratory frequency generator with a sinusoidal wave. For programming the SRAM content during the automated measurements, we used an Olimex JTAG debugger (MSP430-JTAG-TINY-V2), controlled by a Python script using the MSPDebug command line tool [46]. During the whole LLSI measurement, the debugger is left connected and switched on. Fig. 5 showcases images of the SRAM area captured using our setup.

5.3.3 Intel Cyclone IV Registers

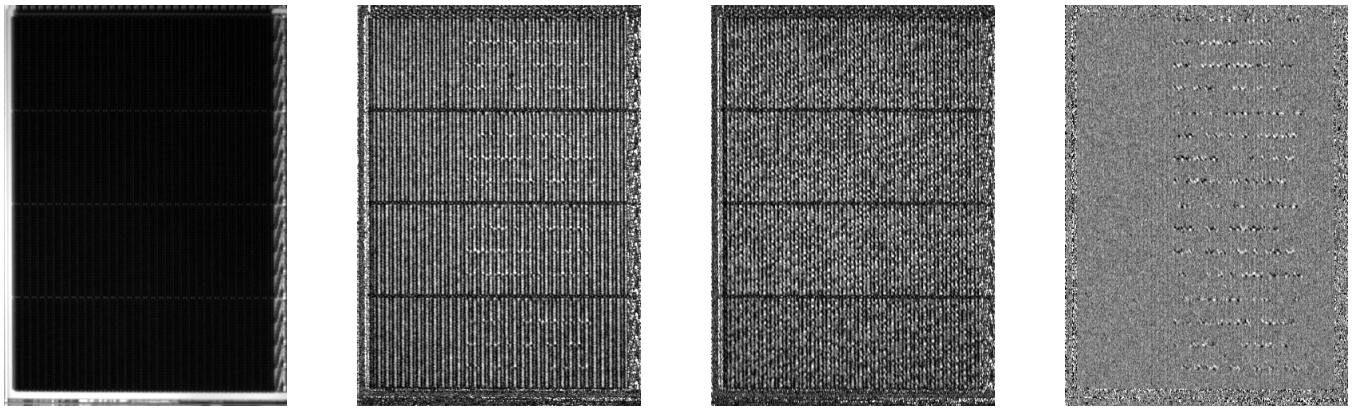
As the third target, we chose the registers of a Intel Cyclone IV FPGA. The FPGA consists of 392 identical logic array blocks (LABs), each comprised of 16 logic elements (LEs), whereas every LE contains one register cell. The chip is manufactured in a 60 nm technology. We had to open the package and solder the chip backside-up on a custom PCB for accessing the chip's backside. To modulate the supply voltage for conducting LLSI, we used a voltage regulator (TI TPS7A7001) and modulated its feedback path with a sinusoidal wave. We created a logic design that updates the register values when applying an external clock with precomputed randomly chosen values during the automated measurements.

By subtracting two LLSI images with different data, we found the LAB's area containing the registers. To reduce the measurement time, we covered only that area with the automated measurements. Consequently, one response image contains one logic array block, and therefore 16 registers, see Fig. 6. From the difference images, we could also estimate the memory cell size to around 7 µm × 9 µm.

6 Results

In this section we apply our deep learning based approach on the response images captured with the automated setup. For all experiments, we first reduced the drift – caused by mechanical instabilities of our setup – between the images in the dataset. For this, we calculated the offset between the optical image captured along with each response image and one fixed optical image by means of an elastic transformation using the MATLAB image processing toolbox. Then we applied the transformation to the corresponding response image. For the sake of simplicity, we will in the following refer to the response images only as "images". To encourage others working with our data, we made all images captured in the context of this work available online.[1]

[1] http://dx.doi.org/10.14279/depositonce-11354



(a) Optical image

(b) LLSI image (512 key bits, rest zeroized)

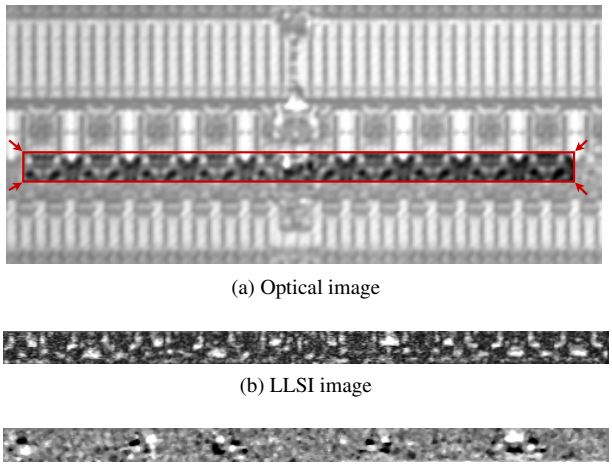
(c) LLSI image (512 key bits, rest randomized)

(d) Difference between two LLSI images (rest zeroized)

Figure 5: Images of the TI MSP430's 1024-byte SRAM area.

Target	# Mem. bits	# Key bits	Technique	Image dimensions	Lens and scanner zoom	# Images	Time/Image (mm:ss)	Total time (hh:mm)
BBRAM	288	256	TLS	985 px × 407 px	50× (×2)	578	02:02	19:35
MSP430 (zeroized)	8192	512	LLSI	503 px × 355 px	50×	433	13:00	93:49
MSP430 (randomized)	8192	512	LLSI	503 px × 355 px	50×	821	13:00	177:53
FPGA Registers	16	16	LLSI	509 px × 28 px	50× (×2)	568	2:40	25:17

Table 1: Overview of devices under test and the captured images in automated measurements.



(a) Optical image

(b) LLSI image

(c) Difference of two LLSI images

Figure 6: Images of one Intel Cyclone IV LAB containing 16 registers.

6.1 Key Extraction from BBRAM

Using the automated setup, we have captured over 500 TLS images of the BBRAM containing randomly chosen keys, see Tab. 1 for details. The memory cells' locations within the image become visible when subtracting two TLS images containing different keys, see Fig. 4c. The relatively large spots indicate that the memory cells cover many pixels, and therefore, we downsized the images with a factor of 0.4 before using them for training. We first investigated if it is possible to extract single key bits from the images (Section 6.1.1). Further, we examined ways for reducing the required time for learning (Section 6.1.2) and the number of images in the training dataset (Section 6.1.3). Finally, we constructed an optimized attack approach from our findings (Section 6.1.4).

6.1.1 Learning single bits

For the first experiments, we fed images containing the entire BBRAM area into the network (cf. Fig. 4). For the CNN, we used a simple VGG-like structure, consisting of just two convolutional layers, followed by a pooling layer, and a FC network with one hidden layer (512 neurons), a dropout layer (rate 0.2), and an output layer with one neuron. For the model summary, see Fig. 18 in the Appendix. For all experiments in this work, we used the Adam optimizer with an initial learning rate of 0.001, binary cross-entropy loss functions, and rectifier activation functions. We randomly split the available images into training (70%), validation (15%), and test (15%) datasets. Further, we applied a batch size of 8 images and set the number of steps per epoch to the number of images in the training dataset divided by the batch size. To deal with the relatively small datasets, we augmented the images by means of an affine transformation with a random rotation of

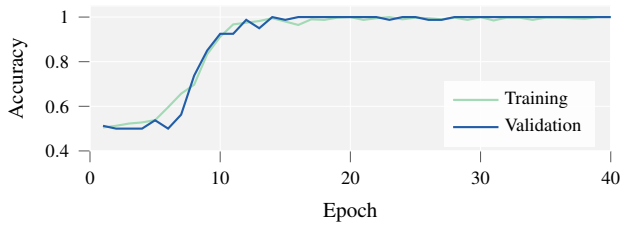


Figure 7: Training and validation accuracy when learning a single bit of the BBRAM key from the full image.

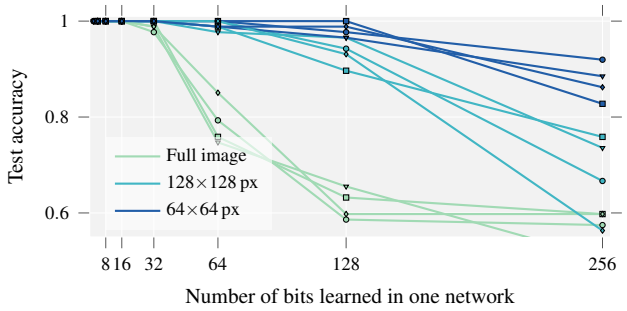


Figure 8: Test accuracies for four bits of the BBRAM key when trying to learn multiple bits in parallel with one network. Shown values depict the maximum out of 3 runs.

2 degrees, a width/height shift of 1 pixel, and a shear of 2 degrees.

The results show that the network can quickly learn one bit of the key, see Fig. 7. We repeated the experiment for 50 randomly chosen bit positions of the key, and recognized, that not all networks lead to a test accuracy of 100%. Therefore, we repeated the network training five times per key bit for different splits of the dataset. In most runs (at least 3 out of 5), we achieved a test accuracy of 100%. The reasons for some networks to perform better and some worse could be the relatively small number of training images and the random initialization of the networks' weights. To make predictions of the secret more reliable, an ensemble learning strategy can potentially be used, for instance, by considering the models from multiple runs in a majority voting fashion. Training a network for one bit took around 180 seconds per run, which – depending on the number of runs – can lead to a training time of some hours to a few days.

6.1.2 Learning bits in parallel

To speed up model training for all key bits, we added more output neurons to the network to learn multiple key bits in parallel. For this, we randomly chose bit positions from the key and checked if we can achieve a simultaneous test accuracy of 100% for all bits. This was the case for up to 4 key bits per network, when training for the same number of epochs as before on the full image, which leads to a 4× speedup in training time. Above 4 bits, the test accuracy was decreasing

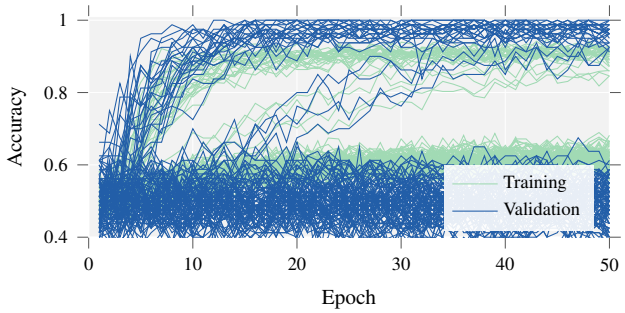


Figure 9: Training history when learning 128 bits of the BBRAM key on a 64 px × 64 px section. The bits contained in the section converge to 100% accuracy, and therefore, can be clearly separated from the others.

significantly. Increasing the number of convolutional and FC layers did not improve the prediction accuracy. Further, we noticed that the achieved performance depends on the spatial distance between the memory cells learned in parallel. When trying to learn cells in close vicinity, the per-bit accuracy is higher than with randomly chosen memory cells.

To further increase the number of bits learned in parallel, we reduced the network's data input dimensions by breaking the images into sections, and training one network for each section. Now not all key bits are contained within one section, and consequently, an accuracy of around 50% might indicate that the section does not contain the corresponding bit. Therefore, we picked four bits that are contained in a specific 128 px × 128 px and 64 px × 64 px section, and tried to learn up to 256 bits of the key in parallel from differently sized sections. The results confirm that a smaller section size leads to a higher accuracy. We could achieve a test accuracy of 100% for all four bits contained in the section when trying to learn up to 32 bits in parallel, see Fig. 8. Although not reaching a very high test accuracy, the network for learning 128 bits in parallel can clearly separate bits that are contained in the section from bits that are not, see Fig. 9. A few bits achieve a higher validation accuracy only in later epochs, presumably because they are not fully contained in the image section, and therefore, are harder to learn.

To sum up, this experiment has shown two things. Firstly, breaking the images into smaller sections can increase the achieved accuracy of the model. Secondly, the bits' rough locations on the image can be found very efficiently, by learning many bit positions of the key in parallel.

6.1.3 Reducing the number of required images

We expect the cost of using the FA microscope, i.e., for capturing the images, to be in orders of magnitude higher than the cost for training the CNNs. Therefore, we consider the required number of training images as the limiting factor regarding the attack costs. Consequently, we tried reduce the

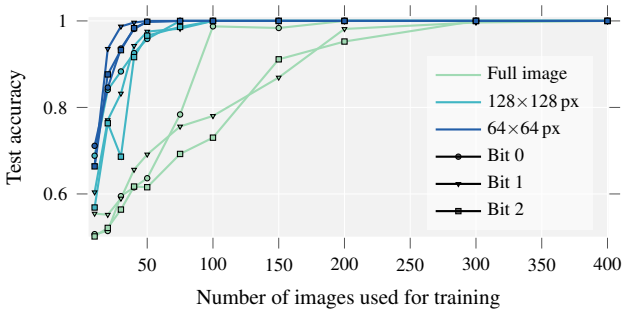


Figure 10: Learning one bit of the BBRAM key per network from differently sized sections, with respect to the number of images used for training. The experiment was repeated for three key bits.

number of samples used for training to a minimum, while still being able to extract the secret. For this, we again tried to learn only single bits per network, and repeated the experiment for three bits of the key on different image section sizes. In a nutshell, the results indicate that training on a smaller section size requires a smaller test dataset, see Fig. 10. Remarkably, to learn a single bit from a 64 px × 64 px section with 100% accuracy demands only 50 training images.

6.1.4 Optimized attack approach

From the above findings, we can now develop an attack approach that is adaptable to constraints like the amount and quality of available images. We propose a two-step divide-and-conquer approach as follows. First, for finding the bits' coarse locations, networks are trained on many bits in parallel for small sections of the original image. Note that high test accuracies are dispensable in this case, since it is only of interest whether a bit is learnable or not. Once the coarse location of each bit is found, networks for each bit (or small groups of bits) can be trained on the corresponding sections.

Localization We chose to reduce the training dataset to only 150 images to better reflect a real attack scenario in which capturing time is expensive, resulting in a dataset acquisition time of 5 hours. We then trained networks for 128 bits in parallel on 64 px × 64 px sections of the images with 5 px overlap at every side, resulting in 21 sections, see Fig. 11. We ran every training three times and selected the most promising section for each bit by first filtering for test accuracies above 75% and then picking the section with the highest number of successful runs. For instance, some of the key bits between 0 and 127 could be learned in section 12, see Fig. 12. The algorithm found bit numbers 0-5, 32-37, 64-69, 96-101, 129, and 131-133[2]. This matches with the memory mapping already discovered in [10]. Note that the bits 128-133 seem to reside directly in the overlap region of sections 12 and 19, and therefore, some bits could be better learned in section 12, and some in 19. We could successfully find the corresponding section for every bit of the key. One training run took 133 seconds, which results in a total localization time of 4:42 hours.

Additionally, we reduced the dataset to 100 images and trained networks only for 64 bits in parallel on 64 px × 64 px sections. The experiment delivered the same localization results as before, with a slightly shorter training time (4:24 hours). Consequently, we believe that tweaks and optimizations can reduce the number of required images even further.

Prediction Once all bits' rough locations are known, at most one network training per key bit is necessary to predict all bits with high accuracy. The previous results indicate that there is a trade-off between training time and training dataset size. Training one network on a 64 px × 64 px image section for one key bit with a dataset consisting of 100 images takes around 30 seconds, resulting in a total training time for all bits of the key of 2:08 hours (for one run per model). To increase the bit prediction accuracy, multiple training runs can potentially be combined in an ensemble learning strategy with only a linear increase in training time.

6.2 Key Extraction from Microcontroller SRAM

On the microcontroller SRAM as our most flexible target, we evaluated two scenarios. In scenario 1 (Section 6.2.1), we programmed a randomly chosen key into 512 bits of the 1 kB (= 8192 bits) memory at the addresses `0x10 – 0x4f`, while keeping the rest of the memory zeroized. This scenario corresponds to the BBRAM target, except for the smaller memory cell sizes and the more distributed memory cells holding the key. In the scenario 2 (Section 6.2.2), the entire memory content is randomized. Again, we consider the same 512 bits of the memory to be the key which should be extracted. This scenario simulates a high amount of irrelevant information in the measurement, caused by other activities on the chip or intended obfuscation.

6.2.1 Scenario 1: Rest zeroized

We captured over 400 images for this scenario, see Tab. 1 for details. Fig. 5d indicates that the images are not as clear as the BBRAM images. The reason is that we did not use an extra 2× scanner zoom like for the BBRAM, because we wanted to fit the whole memory into one image. Furthermore, the memory cells are slightly smaller than those of the BBRAM. The difference image of two different keys (see Fig. 5d) indicates that the key is distributed over large parts of the memory, and therefore, nearly the whole image must be considered for extracting the key bits. We first investigated the required number of images for learning one bit, see Fig. 13. The results show that around 100 images are sufficient to reach 100% test accuracy for a 64 px × 64 px section. For a 128 px × 128 px section, already around 400 images are required to achieve a

[2]Numbering with most significant bit first.

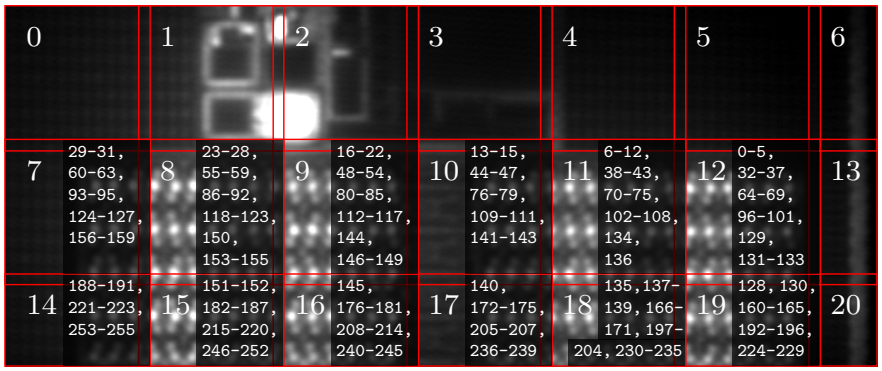


Figure 11: BBRAM memory area split into 64 px × 64 px sections. The small numbers indicate the localized key bits for each section (most significant bit = 0).

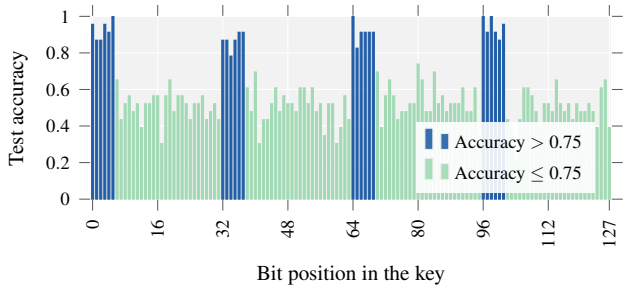


Figure 12: Trying to learn the first 128 bits of the BBRAM key in parallel on section 12 (see Fig. 11) with only 150 images used.

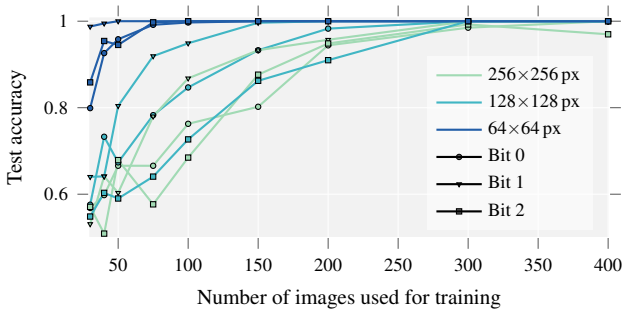


Figure 13: SRAM scenario 1 – Learning one key bit per network from differently sized sections with respect to the number of images used for training. The experiment was repeated for three bit positions.

test accuracy of 100%. The network architecture and setup working best is identical to the setup used for the BBRAM key extraction (Section 6.1.1).

For the localization step, we split the images into 64 px × 64 px sections, resulting in 54 sections, see Fig. 19 in the Appendix. For every section, we trained models on 128 key bits in parallel. We could localize all 512 key bits by using 300 images from the dataset. The results are shown in Tab. 2 in the Appendix. Note that the number of images can be reduced when accepting longer localization times – by learning less bits in paralel.

6.2.2 Scenario 2: Rest randomized

In the previous scenario, there was not much noise present in the images. However, on a real target, surrounding memory cells might not always hold the same value. Therefore, we randomized the entire memory content for scenario 2. The subtraction of two LLSI images shows that no longer any area of interest can be recognized, see Fig. 14. We assumed that this scenario is harder to learn, and therefore, captured over 800 images, see Tab. 1 for details.

As before, we first investigated how many images are required to extract single bits. The results show that – compared to scenario 1 – eight times more images are necessary to achieve a test accuracy of 100%, see Fig. 15. Interestingly, only one of the three bits achieves a test accuracy of 100% for 128 px × 128 px sections (Bit 1). Also for 64 px × 64 px sections, the other two bit positions (Bit 0 and Bit 2) show clearly worse accuracies. For the other bits and larger sections, the number of images seems to be insufficient to achieve a very high test accuracy.

When using 400 images for the localization and learning 128 bits per network, we could map 91% of the bits to the same sections as in scenario 1. In other words, 45 out of 512 bit positions were not found in their correct section. Therefore, we ran the same experiment using 800 images. Although still 12 bit positions were not mapped to the same sections as in scenario 1, they could be located in a neighboring section. The reason is that those bits seem to be located in the overlap region of the two sections. The results show that a high level of irrelevant information increases the amount of required images significantly. Nevertheless, extracting the key is still possible when spending enough time on measurements.

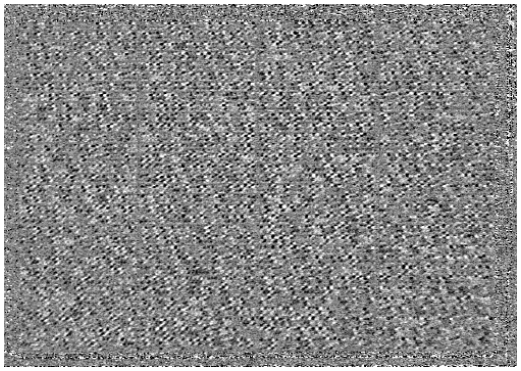


Figure 14: SRAM scenario 2 – Difference between two LLSI images with the entire memory randomized (image rotated clockwise by 90°).

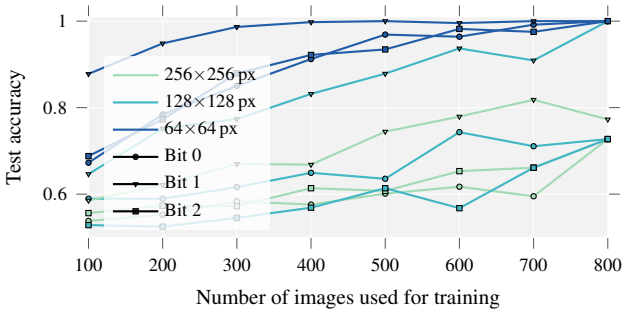


Figure 15: SRAM scenario 2 – Learning one bit per network from differently sized sections while the whole memory content is randomized. Experiment is repeated for three bit positions.

6.3 FPGA Register Content Extraction

Our dataset for this target consists of more than 500 images, each containing one logic array block (LAB) with 16 register bits, see Tab. 1 for details. Note that not all images show the physically same registers on the chip, but instead instances of the same logic layout. Therefore, if the bit values can be learned in our experiment, the resulting predictor can be used to extract data from all LABs distributed over the FPGA.

Like for the other targets, we investigated the influence of the training dataset size on the test accuracy when training networks on a single bit of the secret. The results indicate that – depending on the section size of the images – at most 150 images are required to achieve a test accuracy of 100%, see Fig. 16. Although the bits can already be learned from the full images with a low number of training samples, we further split the images into smaller sections to localize the individual bits in more detail. Fig. 17 shows the results for splitting the images into 8 sections, which already gives very precise information on the bits' position.

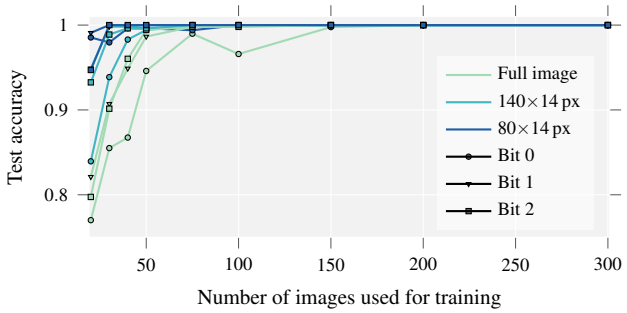


Figure 16: FPGA registers – Learning one bit per network from differently sized sections with respect to the number of images used for training and validation. The experiment was repeated for three bit positions.

7 Discussion

7.1 Scalability of Data Extraction

One important aspect is the scalability of our approach towards the extraction of larger chunks of data and implementations employing classical countermeasures against SCA attacks (e.g., Boolean masking).

In our experiments on the MSP430 microcontroller (Section 6.2), we have captured images of the full 1024-byte SRAM with randomly chosen content. We have defined 64 bytes in a fixed address range as the key bits, and have shown that all bits can be localized and extracted from the images. Since we could have chosen any other address range within the memory as key storage, it will also be possible to extract the entire memory content from the images with only a linear increase in extraction time. Consequently, we expect our approach to work also on larger chunks of data with only a linearly growing effort.

One might ask if the approach is also applicable when the key is not present in plaintext on the chip. Examples for implementations that do not require a key in plaintext are masked versions of cryptographic cores that work on shared forms of the key [47]. Previous work has already shown that all key shares can be extracted using laser-assisted SCA when all potential memory/register locations are known to the adversary: either by direct readout or with the help of a SAT solver [11]. In this work, we assume zero knowledge about the memory locations on the chip.

In preliminary experiments, we presume a 2-share Boolean masking of the key, meaning that the unmasked key can only be obtained by XOR'ing two values stored in the memory. We artificially created the masking on the available dataset by defining pairs of memory locations as the shares. In other words, on a memory snapshot containing N bit values $b_0 \dots b_{N-1}$, one key bit k for a 2-share masking is $k = b_x \oplus b_y$ $(0 \leq x, y < N, x \neq y)$. During the profiling phase, only the unmasked key k is known to the adversary. We

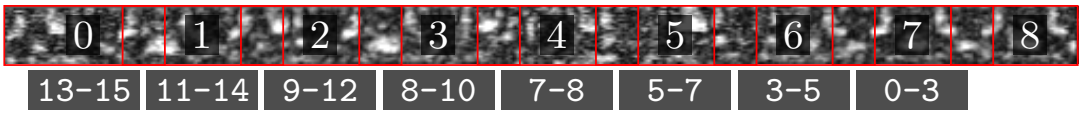


Figure 17: Sections of the FPGA register area for localizing the bits' rough positions. The number ranges indicate the bit positions of the secret localized in the respective section.

trained models on a 128 px × 128 px section of the BBRAM images containing $N = 47$ bits, and achieved 100% test accuracy for all exemplarily tested bit combinations (e.g., for $(x,y) \in \{(0,4),(1,8),(32,66)\}$, cf. Fig. 11).

Hence, the network has not only learned the memory locations of the individual shares, but also that the values have to be XOR'ed to obtain the unmasked key. We used the same neural network structure as in all the other experiments presented in this work and observed that the model needs to be trained for more epochs than for the direct key extraction. On the MSP430 microcontroller SRAM, we only had success on some bit combinations, and therefore, we believe that the network architecture will have to be adapted to work more reliably. A more thorough exploration of masked data extraction can be conducted in the future using the data collected in this work. In summary, also the unmasked key of a masked implementation can be extracted using our laser-assisted SCA approach.

7.2 Optical Resolution and Cell Size

Optical resolution is defined as the ability of an optical system to differentiate between two closely spaced objects. Because of constant decrease in feature sizes – now reaching down to the 5 nm node, optical resolution has been a growing concern for the FA community. Debugging the root cause of a failure can require to resolve adjacent minimum size transistors from each other, which might be challenging when we think of the most recent technology nodes. Tools such as the solid immersion lens (SIL) and visible light source systems [48, 49] have been introduced to overcome this problem. It has been shown that a SIL can improve the optical resolution down to approximately 200 nm, enabling optical probing even for 10 nm technology nodes [17, 50]. With our setup, we can achieve a laser spot diameter of approximately 1 µm without a SIL. Laser power at the center of the spot is the strongest and decreases exponentially through the edge.

The transistors in the SRAM cells are often designed to be larger than those used in the logic part of the chip to avoid off-leakage current related data loss. Although the DUTs in our experiments were manufactured in technology nodes down to 20 nm, the contained memory cells were larger than expected. Among the DUTs that we have used, the smallest cell size is 2.5 µm × 1.9 µm in MSP430 which is still larger than the 1 µm laser diameter. In the case of the Xilinx Ultrascale BBRAM, the cell size is even larger, although the technology size is much smaller. This shows that cell sizes do not always proportionally scale with the technology nodes, but cell size scaling also depends on many other parameters such as current leakage or supply voltage. The designers have to keep the transistor sizes bigger to maintain the circuit performance and the yield. In addition to that, while logic density continues to double in every technology generation, the memory cell size shrink cannot keep up the trend at the same pace. As a result, the memory density increase remains less than double at every new technology node [51]. The limiting factor appears to be lithography and the cost associated with it [52].

The question whether it is possible to extract logic states from memory cells that are smaller than the laser spot size can not be answered trivially. While for FA purposes it might be important to target only a single transistor, for our approach it is only important that the response image differs in some way between the logic states `0` and `1`. As a matter of fact, the distances and the positions of the opposite state transistors with respect to each other are more important than the transistor sizes. For our DUTs, we do not know the exact memory structure, and we have not tested our approach on memories other than presented in this work. However, this is among our future research interests.

For the optical SCA techniques used in this work, the laser beam is scanned over the device pixel-wise. When reducing the pixel size to values smaller than the laser spot diameter, for every pixel the superimposed signal/response originating from multiple transistors or memory cells will be captured. Consequently, the resulting response image will be noisy. We suppose that image processing tools like CNNs can be used to recover the logic state from the interfering signals. To the best of our knowledge, this has not been investigated in the hardware security community, and therefore, it is among our planned future works. In conclusion, the optical resolution might be a challenge when going to memories with smaller cell sizes and higher cell densities. However, we assume that optical contactless probing will continue to be present for a while due to the reasons mentioned above.

7.3 Chip Access

All the above mentioned SCA techniques are performed through the chip backside, which means that the attacker should have access to the bulk silicon. Since many modern ICs are manufactured in flip-chip packages, optical attacks are easy to conduct and often even do not require extra prepa-

ration steps. For instance, the Xilinx Kintex Ultrascale FPGA is shipped in a bare-die flip-chip package, and therefore, no preparation was needed for silicon access. In contrast, the packages of the other targeted devices had to be opened and soldered back-side up on a custom PCB for accessing the backside, which makes it a semi-invasive attack. Nevertheless, it should be noted that for technology nodes of 20 nm and below, flip-chip packages are becoming more prevalent due to performance, size and cost issues [53].

7.4 Attack Cost and Time Expenditure

Our investigations have shown that – depending on the area of interest on the chip and the imaging resolution – several hours to a few days have to be spent for automated measurements on the training device. This time is presumably the most costly period when conducting the proposed attack. This is not a challenge when the attacker owns a setup for conducting the attacks. The tools for conducting TLS and LLSI cost around $1M, whereas a setup for conducting only TLS can be acquired for around $100k [23]. Since a laser scanning microscope is common equipment in FA labs around the globe, a suitable setup can also be rented for about 300$/h including an operator. Consequently, we can calculate the costs for acquiring the images as given in Tab. 1 to $509 for the BBRAM (50 images), around $6.5k for scenario 1 (100 images) and $52k for scenario 2 on the microcontroller SRAM, and $667 for the FPGA registers (50 images). Note that due to the mostly automated measurements, which can also run unsupervised during the night, those fares could presumably be reduced. Furthermore, a more stable optical setup would avoid the need for a frequent auto-focus and drift correction, and therefore, can potentially reduce the measurement times according to our estimations by up to 50%. Although we agree that the costs are still high for some scenarios, we would like to stress that the gathered model is applicable to all devices of a device series, and can extract the secrets contained in multiple devices.

7.5 Key Control

One might argue that it is not always true that the adversary can program different keys into the NVM on a training device, for instance, when one-time programmable (OTP) memories like e-fuses or ROMs are used. We admit that such keys cannot be extracted using our approach. However, in many applications a OTP memory only stores a key-decryption-key, which is used to decrypt other application keys contained in reprogrammable NVMs. This makes the system more flexible and keys can be updated together with the device's firmware. Since the application keys will be decrypted by some cryptographic core on the device, they will in the end also be stored in registers on the chip. We have shown that this kind of application keys can be targeted using our approach.

7.6 Potential Countermeasures

When looking for potential countermeasures, one should keep in mind that potentially many different FA techniques can be used to read out the logic states of the device under attack. Therefore, a countermeasure should at best protects against all possible attack techniques. In other words, there exist various countermeasures that are effective against some FA-based attack techniques, but do not necessarily prevent other methods.

One technique proposed for protecting semiconductor intellectual property is IC camouflaging [54, 55]. Therefore, one might ask if camouflaging also can protect against memory readout. The idea behind camouflaging is to insert logic gates whose functionality cannot be extracted by delayering the chip and applying imaging techniques like SEM. However, since optical techniques rely on interactions with the actual transistors, they can still recognize the function of the camouflaged gates [54]. In other words, it would be possible to extract the logic states using activity maps of the circuit. Consequently, camouflaging does not seem to be an appropriate countermeasure.

The foremost requirement for our attack approach to succeed is access through the chip's backside. Active backside coatings [56] can prevent the optical access to the chip's silicon by adding an opaque coating layer. By actively checking the intactness of the coating, attempts to remove it can be detected. Since removing the silicon substrate from the chip backside is necessary for conducting SEM- and FIB-based attacks, an active coating can also help in these cases. However, to the best of our knowledge, there is no implementation of an active backside coating ready for mass production.

According to the preliminary results presented in Section 7.1 on masking implementations, Boolean masking seems to increase the effort for the attacker, but does not prevent laser-assisted SCA to a sufficient degree.

8 Conclusion

Hardware attacks using sophisticated FA tools are often seen as too costly and time-consuming to pose a severe threat to modern ICs and SoCs. Therefore, vendors usually rely on the complexity of the layout and tamper-proof memories to prevent key extraction. However, for being used, every key will be cached into memory cells that are vulnerable to probing techniques, such as optical SCA. In this work, we have shown that the automation of FA tools combined with deep learning techniques reduces the required effort by an adversary significantly. We carried out highly automated measurements on three different hardware targets holding an attacker-controlled secret in their memories. Besides, we have demonstrated how to fully extract the secret from the captured images without knowing the chip's layout, especially the memory cells' design, geometry, and exact location. We believe that our ap-

proach has the potential to antiquate the expensive reverse engineering part of hardware attacks by offering a very targeted and generic procedure for key extraction, which can also be applied in the presence of largely irrelevant information and activities on the chip. Hence, a great deal of attention has to be paid to this threat when designing new RoT devices for critical applications. While, in this work, we presented an offensive application of our approach, it also can be utilized to assess the vulnerability of the products in the early stages of the design, and consequently, assist in finding the right defense techniques.

Acknowledgment

The work described in this paper has been supported in part by the Einstein Foundation in form of an Einstein professorship – EP-2018-480, in part by the Deutsche Forschungsgemeinschaft (DFG – German Research Foundation) under the priority programme SPP 2253 – 422730034, and in part by the German Ministry for Education and Research as BIFOLD – Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A). The authors would also like to acknowledge Hamamatsu Photonics K.K. Japan and Germany for their help and support on the PHEMOS system.

References

[1] TechInsights Inc. *Semiconductor Analysis & IP Services*. 2020. URL: https://www.techinsights.com/.

[2] Texplained. *Hardware Security Insight*. 2021. URL: https://www.texplained.com/.

[3] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor. "A Survey on Chip to System Reverse Engineering". In: *ACM Journal on Emerging Technologies in Computing Systems* 13.1 (2016), 6:1–6:34. DOI: 10.1145/2755563.

[4] C. Kison, J. Frinken, and C. Paar. "Finding the AES Bits in the Haystack: Reverse Engineering and SCA Using Voltage Contrast". In: *Cryptographic Hardware and Embedded Systems – CHES 2015*. Springer, 2015, pp. 641–660. DOI: 10.1007/978-3-662-48324-4_32.

[5] F. Courbon, S. Skorobogatov, and C. Woods. "Reverse Engineering Flash EEPROM Memories Using Scanning Electron Microscopy". In: *International Conference on Smart Card Research and Advanced Applications*. Springer, 2017, pp. 57–72. DOI: 10.17863/CAM.7164.

[6] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert. "Breaking and Entering Through the Silicon". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013, pp. 733–744. DOI: 10.1145/2508859.2516717.

[7] O. Kömmerling and M. G. Kuhn. "Design Principles for Tamper-Resistant Smartcard Processors". In: *Proceedings of the USENIX Workshop on Smartcard Technology (WOST'99)*. USENIX Association, 1999.

[8] H. Lohrke, S. Tajik, C. Boit, and J.-P. Seifert. "No Place to Hide: Contactless Probing of Secret Data on FPGAs". In: *Cryptographic Hardware and Embedded Systems – CHES 2016*. Springer, 2016, pp. 147–167. DOI: 10.1007/978-3-662-53140-2_8.

[9] S. Tajik, H. Lohrke, J.-P. Seifert, and C. Boit. "On the Power of Optical Contactless Probing: Attacking Bitstream Encryption of FPGAs". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 1661–1674. DOI: 10.1145/3133956.3134039.

[10] H. Lohrke, S. Tajik, T. Krachenfels, C. Boit, and J.-P. Seifert. "Key Extraction Using Thermal Laser Stimulation". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 573–595. DOI: 10.13154/tches.v2018.i3.573-595.

[11] T. Krachenfels, F. Ganji, A. Moradi, S. Tajik, and J.-P. Seifert. *Real-World Snapshots vs. Theory: Questioning the t-Probing Security Model*. 2020. arXiv: 2009.04263 [cs].

[12] H. Maghrebi, T. Portigliatti, and E. Prouff. "Breaking Cryptographic Implementations Using Deep Learning Techniques". In: *Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 3–26. DOI: 10.1007/978-3-319-49445-6_1.

[13] 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*. 2018. URL: https://eprint.iacr.org/2018/053.

[14] T. Kubota, K. Yoshida, M. Shiozaki, and T. Fujino. "Deep Learning Side-Channel Attack Against Hardware Implementations of AES". In: *2019 22nd Euromicro Conference on Digital System Design (DSD)*. 2019, pp. 261–268. DOI: 10.1109/DSD.2019.00046.

[15] S. R. Hou, Y. J. Zhou, and H. M. Liu. "Convolutional Neural Networks for Profiled Side-Channel Analysis". In: *Radioengineering* 27.3 (2019), pp. 651–658. DOI: 10.13164/re.2019.0651.

[16] M. T. Rahman, S. Tajik, M. S. Rahman, M. Tehranipoor, and N. Asadizanjani. "The Key Is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes". In: *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 2020.

[17] C. Boit, S. Tajik, P. Scholz, E. Amini, A. Beyreuther, H. Lohrke, and J. P. Seifert. "From IC Debug to Hardware Security Risk: The Power of Backside Access and Optical Interaction". In: *Proceedings of the 23rd International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*. IEEE, 2016, pp. 365–369. DOI: 10.1109/IPFA.2016.7564318.

[18] S. Tajik, D. Nedospasov, C. Helfmeier, J.-P. Seifert, and C. Boit. "Emission Analysis of Hardware Implementations". In: *17th Euromicro Conference on Digital System Design*. IEEE, 2014, pp. 528–534. DOI: 10.1109/DSD.2014.64.

[19] D. Nedospasov, J. P. Seifert, C. Helfmeier, and C. Boit. "Invasive PUF Analysis". In: *Proceedings of the 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 2013, pp. 30–38. DOI: 10.1109/FDTC.2013.19.

[20] B. Niu, G. M. E. Khoo, Y.-C. S. Chen, F. Chapman, D. Bockelman, and T. Tong. "Laser Logic State Imaging (LLSI)". In: *Proceedings from the 40th International Symposium for Testing and Failure Analysis (ISTFA 2014)*. ASM International, 2014, p. 65.

[21] T. H. Geballe and G. W. Hull. "Seebeck Effect in Silicon". In: *Physical Review* 98.4 (1955), pp. 940–947. DOI: 10.1103/PhysRev.98.940.

[22] T. Kiyan, H. Lohrke, and C. Boit. "Comparative Assessment of Optical Techniques for Semi-Invasive SRAM Data Read-out on an MSP430 Microcontroller". In: *ISTFA 2018: Proceedings from the 44th International Symposium for Testing and Failure Analysis*. ASM International, 2018, p. 266.

[23] T. Krachenfels, H. Lohrke, J.-P. Seifert, E. Dietz, S. Frohmann, and H.-W. Hübers. "Evaluation of Low-Cost Thermal Laser Stimulation for Data Extraction and Key Readout". In: *Journal of Hardware and Systems Security* 4.1 (2020), pp. 24–33. DOI: 10.1007/s41635-019-00083-9.

[24] W. M. Yee, M. Paniccia, T. Eiles, and V. Rao. "Laser Voltage Probe (LVP): A Novel Optical Probing Technology for Flip-Chip Packaged Microprocessors". In: *Proceedings of the 1999 7th International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*. 1999, pp. 15–20. DOI: 10.1109/IPFA.1999.791222.

[25] H. Zhang, P. Tian, X. Qian, and W. Wang. "Electro Optical Probing / Frequency Mapping (EOP/EOFM) Application in Failure Isolation of Advanced Analogue Devices". In: *2017 IEEE 24th International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*. 2017. DOI: 10.1109/IPFA.2017.8060131.

[26] T. Moos. "Static Power SCA of Sub-100 Nm CMOS ASICs and the Insecurity of Masking Schemes in Low-Noise Environments". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), pp. 202–232. DOI: 10.13154/tches.v2019.i3.202-232.

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[29] Y. LeCun, F. J. Huang, and L. Bottou. "Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004.* Vol. 2. 2004. DOI: 10.1109/CVPR.2004.1315150.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs].

[31] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. "Deep Learning for Generic Object Detection: A Survey". In: *International Journal of Computer Vision* 128.2 (2020), pp. 261–318. DOI: 10.1007/s11263-019-01247-4.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. 2012, pp. 1106–1114. DOI: 10.1145/3065386.

[33] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[34] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. *Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors*. 2012. arXiv: 1207.0580 [cs].

[35] P. Simard, D. Steinkraus, and J. Platt. "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis". In: *Seventh International Conference on Document Analysis and Recognition*. 2003, pp. 958–963. DOI: 10.1109/ICDAR.2003.1227801.

[36] L. Lerman, R. Poussier, G. Bontempi, O. Markowitch, and F.-X. Standaert. "Template Attacks vs. Machine Learning Revisited (and the Curse of Dimensionality in Side-Channel Analysis)". In: *Constructive Side-Channel Analysis and Secure Design*. Springer International Publishing, 2015, pp. 20–33. DOI: 10.1007/978-3-319-21476-4_2.

[37] F.-X. Standaert, F. Koeune, and W. Schindler. "How to Compare Profiled Side-Channel Attacks?" In: *Applied Cryptography and Network Security*. Springer, 2009, pp. 485–498. DOI: 10.1007/978-3-642-01957-9_30.

[38] O. Choudary and M. G. Kuhn. "Template Attacks on Different Devices". In: *Constructive Side-Channel Analysis and Secure Design*. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 179–198. DOI: 10.1007/978-3-319-10175-0_13.

[39] R. Elnaggar and K. Chakrabarty. "Machine Learning for Hardware Security: Opportunities and Risks". In: *Journal of Electronic Testing* 34.2 (Apr. 1, 2018), pp. 183–201. DOI: 10.1007/s10836-018-5726-9.

[40] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle. "Machine Learning in Side-Channel Analysis: A First Study". In: *Journal of Cryptographic Engineering* 1.4 (2011), p. 293. DOI: 10.1007/s13389-011-0023-x.

[41] K. Hasegawa, M. Yanagisawa, and N. Togawa. "Hardware Trojans Classification for Gate-Level Netlists Using Multi-Layer Neural Networks". In: *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. 2017, pp. 227–232. DOI: 10.1109/IOLTS.2017.8046227.

[42] M. Chen and P. Liu. *Deep Learning-Based FPGA Function Block Detection Method Using an Image-Coded Representation of Bitstream*. July 20, 2020. arXiv: 2007.11434 [cs, eess].

[43] C. Boit, T. Kiyan, T. Krachenfels, and J.-P. Seifert. "Logic State Imaging From FA Techniques for Special Applications to One of the Most Powerful Hardware Security Side-Channel Threats". In: *2020 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*. 2020, pp. 1–7. DOI: 10.1109/IPFA49335.2020.9261000.

[44] Texas Instruments Inc. *MSP-PMBUS PMBus Software Library for MSP MCUs*. Version 1.0. 2015. URL: https://www.ti.com/tool/MSP-PMBUS.

[45] Xilinx Inc. *Vivado Design Suite Tcl Command Reference Guide (UG835)*. 2019.

[46] D. Beer. *Dlbeer/Mspdebug*. 2020. URL: https://github.com/dlbeer/mspdebug.

[47] Y. Ishai, A. Sahai, and D. A. Wagner. "Private Circuits: Securing Hardware against Probing Attacks". In: *Advances in Cryptology - CRYPTO 2003*. Vol. 2729. LNCS. Springer, 2003, pp. 463–481. DOI: 10.1007/978-3-540-45146-4_27.

[48] J. Beutler, V. C. Hodges, J. J. Clement, J. Stevens, E. I. C. Jr, S. Silverman, and R. Chivas. *Visible Light LVP on Bulk Silicon Devices*. Tech. rep. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2015.

[49] C. Boit, H. Lohrke, P. Scholz, A. Beyreuther, U. Kerst, and Y. Iwaki. "Contactless Visible Light Probing for Nanoscale ICs through 10 μm Bulk Silicon". In: *Proceedings of the 35th Annual NANO Testing Symposium (NANOTS 2015)*. 2015, pp. 215–221.

[50] M. Von Haartman, S. Rahman, S. Ganguly, J. Verma, A. Umair, and T. Deborde. "Optical Fault Isolation and Nanoprobing Techniques for the 10 Nm Technology Node and Beyond". In: *Proceedings of the 41st International Symposium for Testing and Failure Analysis*. 2015, pp. 47–51.

[51] D. Maheshwari. "6.1 Memory and System Architecture for 400Gb/s Networking and Beyond". In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 2014, pp. 116–117. DOI: 10.1109/ISSCC.2014.6757362.

[52] A. Keshavarzi, D. Maheshwari, D. Mattos, R. Kapre, S. Krishnegowda, M. Whately, and S. Gopalswamy. "Directions in Future of SRAM with QDR-WideIO for High Performance Networking Applications and Beyond". In: *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*. 2014, pp. 1–6. DOI: 10.1109/CICC.2014.6946029.

[53] H. Tong, Y. Lai, and C. Wong. *Advanced Flip Chip Packaging*. Springer US, 2013.

[54] B. Shakya, H. Shen, M. Tehranipoor, and D. Forte. "Covert Gates: Protecting Integrated Circuits with Undetectable Camouflaging". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(3)* (2019), pp. 86–118. DOI: 10.13154/tches.v2019.i3.86-118.

[55] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri. "Security analysis of integrated circuit camouflaging". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 709–720. DOI: 10.1145/2508859.2516656.

[56] E. Amini, A. Beyreuther, N. Herfurth, A. Steigert, B. Szyszka, and C. Boit. "Assessment of a Chip Backside Protection". In: *Journal of Hardware and Systems Security* 2.4 (2018), pp. 345–352. DOI: 10.1007/s41635-018-0052-3.

Appendix

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputImage (InputLayer)      [(None, 158, 384, 1)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 158, 384, 32)      320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 158, 384, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 79, 192, 32)       0
_________________________________________________________________
flatten (Flatten)            (None, 485376)            0
_________________________________________________________________
dense (Dense)                (None, 512)               248513024
_________________________________________________________________
activation (Activation)      (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
outputBit079 (Dense)         (None, 1)                 513
=================================================================
Total params: 248,523,105
Trainable params: 248,523,105
Non-trainable params: 0
_________________________________________________________________
```

Figure 18: CNN model summary for the BBRAM experiments, here for learning bit 79 of the key.

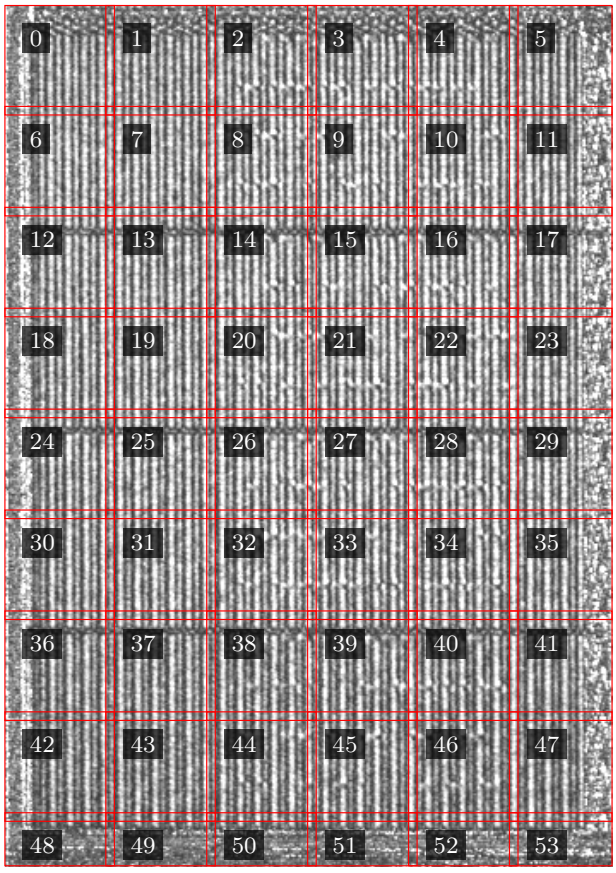


Figure 19: Sections of the MSP430's SRAM area used for localizing the bits.

Section	**Bit positions** (most significant bit first)
2	375, 383, 391, 399, 407, 415, 423, 431, 439, 447, 455, 463, 471, 479, 487, 495, 503, 511
3	183, 191, 199, 207, 215, 223, 231, 239, 247, 255, 263, 271, 279, 287, 295, 303, 311, 319, 327, 335, 343, 351, 359, 367
4	7, 15, 23, 31, 39, 47, 55, 63, 71, 79, 87, 95, 103, 111, 119, 127, 135, 143, 151, 159, 167, 175
8	374, 382, 390, 398, 406, 414, 422, 430, 438, 446, 454, 462, 470, 478, 486, 494, 502, 510
9	182, 198, 206, 214, 222, 230, 238, 246, 254, 262, 270, 278, 286, 294, 302, 310, 318, 326, 334, 342, 350, 358, 366
10	6, 14, 22, 30, 38, 46, 54, 62, 70, 78, 86, 94, 102, 110, 118, 126, 134, 142, 150, 158, 166, 174, 190
14	373, 381, 389, 397, 405, 413, 421, 429, 437, 445, 453, 461, 469, 477, 485, 493, 501, 509
15	181, 189, 197, 205, 213, 221, 229, 237, 245, 253, 261, 269, 277, 285, 293, 301, 309, 317, 325, 333, 341, 349, 357, 365
16	5, 13, 21, 29, 37, 45, 53, 61, 69, 77, 85, 93, 101, 109, 117, 125, 133, 141, 149, 157, 165, 173
20	372, 380, 388, 396, 404, 412, 420, 428, 436, 444, 452, 460, 468, 476, 484, 492, 500, 508
21	188, 196, 204, 212, 220, 228, 236, 244, 252, 260, 268, 276, 284, 292, 300, 308, 316, 324, 332, 340, 348, 356, 364
22	4, 12, 20, 28, 36, 44, 52, 60, 68, 76, 84, 92, 100, 108, 116, 124, 132, 140, 148, 156, 164, 172, 180
26	371, 387, 395, 403, 411, 419, 427, 435, 443, 451, 459, 467, 475, 483, 491, 499, 507
27	179, 195, 203, 211, 219, 227, 235, 243, 251, 259, 267, 275, 283, 291, 299, 307, 315, 323, 331, 339, 347, 355, 363, 379
28	3, 11, 19, 27, 35, 43, 51, 59, 67, 75, 83, 91, 99, 107, 115, 123, 131, 139, 147, 155, 163, 171, 187
32	370, 378, 386, 394, 402, 410, 418, 426, 434, 442, 450, 458, 466, 474, 482, 490, 498, 506
33	178, 186, 194, 202, 210, 218, 226, 234, 242, 250, 258, 266, 274, 282, 290, 298, 306, 314, 322, 330, 338, 346, 354, 362
34	2, 10, 18, 26, 34, 42, 50, 58, 66, 74, 82, 90, 98, 106, 114, 122, 130, 138, 146, 154, 162, 170
38	369, 377, 385, 393, 401, 409, 417, 425, 433, 441, 449, 457, 465, 473, 481, 489, 497, 505
39	185, 193, 201, 209, 217, 225, 233, 241, 249, 257, 265, 273, 281, 289, 297, 305, 313, 321, 329, 337, 345, 353, 361
40	1, 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 105, 113, 121, 129, 137, 145, 153, 161, 169, 177
44	368, 376, 384, 392, 400, 408, 416, 424, 432, 440, 448, 456, 464, 472, 480, 488, 496, 504
45	192, 200, 208, 216, 224, 232, 240, 248, 256, 264, 272, 280, 288, 296, 304, 312, 320, 328, 336, 344, 352, 360
46	0, 8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96, 104, 112, 120, 128, 136, 144, 152, 160, 168, 176, 184

Table 2: Localization of the key in the MSP430's SRAM for the sections shown in Fig. 19.