Practical Operation Extraction from Electromagnetic Leakage for Side-Channel Analysis and Reverse Engineering

Pieter Robyns
Hasselt University - tUL - Expertise Centre for Digital Media
Hasselt, Belgium
pieter.robyns@uhasselt.be

Mariano Di Martino
Hasselt University - tUL - Expertise Centre for Digital Media
Hasselt, Belgium
mariano.dimartino@uhasselt.be

Dennis Giese
Northeastern University
Boston, Massachusetts, USA
dgiese@ccs.neu.edu

Wim Lamotte
Hasselt University - tUL - Expertise Centre for Digital Media
Hasselt, Belgium
wim.lamotte@uhasselt.be

Peter Quax
Hasselt University - tUL - Flanders Make - EDM
Hasselt, Belgium
peter.quax@uhasselt.be

Guevara Noubir
Northeastern University
Boston, Massachusetts, USA
g.noubir@northeastern.edu

ABSTRACT

Determining which operations are being executed by a black-box device is an important challenge to tackle in reverse engineering. Furthermore, in order to perform a successful side-channel analysis (SCA) of said operations, their precise timing must be determined. In this paper, we tackle these two challenges in context of an electromagnetic (EM) analysis of a NodeMCU Amica IoT device. More specifically, we propose a convolutional neural network (CNN) architecture that is designed to classify operations performed by the NodeMCU out of a set of 8 possible operations, namely OpenSSL AES, native AES, TinyAES, OpenSSL DES, SHA1-PRF, HMAC-SHA1, SHA1, and SHA1Transform. In addition, we use the same architecture to predict the start and end times of the operation, thereby removing the need for firmware modifications or manual triggers in SCA. Our approach is evaluated using a 66 GB dataset containing 69,632 complex traces of EM leakage, captured with a USRP B210 software defined radio. The best variant of our methodology achieves a classification accuracy of 96.47%, and is able to predict the start and end times of the operation within 34 µs of the ground truth on average. We compare our methodology to classical template matching, and provide our open-source implementation and datasets to the community so that the achieved results can be reproduced.

CCS CONCEPTS

• **Security and privacy** → **Hardware reverse engineering**; **Cryptanalysis and other attacks**; • **Computing methodologies** → *Neural networks*.

KEYWORDS

electromagnetic leakage, side channels, privacy, reverse engineering, Wi-Fi, Internet of Things, neural networks, fingerprinting

ACM Reference Format:
Pieter Robyns, Mariano Di Martino, Dennis Giese, Wim Lamotte, Peter Quax, and Guevara Noubir. 2020. Practical Operation Extraction from Electromagnetic Leakage for Side-Channel Analysis and Reverse Engineering. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20), July 8–10, 2020, Linz (Virtual Event), Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3395351.3399362

1 INTRODUCTION

In Side-Channel Analysis (SCA), physical effects caused by the implementation of a (cryptographic) operation are analyzed in order to determine whether or how they leak information to an adversary [39]. Examples of such physical effects that can leak information include timing [16], temperature [4], electromagnetic (EM) radiation [11, 28], power consumption [15, 25], and sound [32]. Particularly in EM analysis, the EM radiation emitted by a device during computations is considered. Previous works have shown that such radiation can indeed leak information about, for example, the key being used in a cryptographic system [12].

To successfully perform an EM analysis, several challenges must be tacked: (i) the adversary must correctly position a probe near a leaking region of interest, (ii) the adversary must measure the EM emanations of the system at an adequate sampling rate, (iii) the operation of interest must be localized in the performed measurement using e.g. a trigger [5, 7, 22, 34], (iv) the measurements must be aligned in time, for example by applying a cross-correlation or Sum of Differences (SOD) metric [5, 23, 40], and (v) the measurements must be processed and attacked using a methodology that is tailored to the targeted cipher. For instance, a Simple Electromagnetic Analysis (SEMA) is generally applied to algorithms such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA), whereas Differential Electromagnetic Analysis (DEMA) and Correlation Electromagnetic Analysis (CEMA) attacks are common for algorithms such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES). The adversary must therefore know which algorithm is performed on

WiSec '20, July 8–10, 2020, Linz (Virtual Event), Austria
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8006-5/20/07...$15.00
https://doi.org/10.1145/3395351.3399362

the device. In a black-box attack scenario, this is generally not the case, since the adversary can only observe the inputs and outputs. For instance, in context of block ciphers, note that the output of a secure block cipher should be indistinguishable from other block ciphers of the same block size.

In this paper, we propose an approach that can be applied in practice to solve two of the challenges presented above. More specifically, we propose a methodology to (i) identify which operations are performed by a device, based only on the EM radiation it emits, and (ii) to localize the start and end times of this algorithm within the EM trace. Our methodology thereby allows to extract cryptographic operations for EM analysis without requiring firmware modifications that incorporate a "trigger" for the operations, and to identify the usage of potentially vulnerable implementations in black-box devices under test. For instance, by probing a black-box device during the execution of an unknown hashing function, our methodology could be applied to determine that SHA-1 is being used, which is considered vulnerable [19].

To achieve this, we developed a deep Convolutional Neural Network (CNN) architecture based on the SCA architecture proposed by Benadjila et al. [27]. Our architecture supports long input EM traces (131,072 samples), and is able to predict classes of operations along with their start and end times in the trace in real time. Our methodology can thus be applied in the domain of SCA as well as the domain of forensics, e.g. when the operations performed by a black-box device must be reverse engineered [25].

Our methodology is compared to template matching using a training dataset of 65,536 random operations and test dataset of 4,096 random operations performed on a NodeMCU Amica device. This dataset was captured with a USRP B210 and is made publicly available. A single random operation in the dataset can come from the following set of possibilities: OpenSSL AES, native AES, TinyAES, OpenSSL DES, SHA1-PRF, HMAC-SHA1, SHA1, SHA1-Transform and no operation. These operations are commonly implemented on Internet-of-Things (IoT) devices. For example, SHA1-PRF is typically called during a WPA2 4-Way Handshake. Finally, we perform an experiment where the EM emissions of the NodeMCU are captured while it is connecting to an Access Point (AP).

In summary, our paper brings the following novel contributions:

- We provide a 66 GB dataset containing 69,632 complex traces of EM emissions from the above operations, performed at random on a NodeMCU device. Labeled bounding boxes are provided for 768 of these traces. In addition, the dataset includes 3 full EM traces of the NodeMCU performing a Wi-Fi connection procedure to a `hostapd` AP.
- We propose a CNN architecture to classify which random operation was performed by the NodeMCU based on a single EM leakage measurement. Furthermore, our CNN simultaneously predicts the location of the operation within the trace with high accuracy. We show that we can identify different implementations of the same algorithm (e.g. OpenSSL AES vs TinyAES), and that this classification can be performed in real time. Furthermore, we open source the tools used for both capture and analysis.
- We evaluate our methodolgy on the acquired datasets and compare to classic instananeous amplitude-based and Short-Time Fourier Transform (STFT)-based template matching. The rationale for selecting these two features as a basis for template matching will be explained in Section 3.5.

The structure of this paper is as follows: in Section 2, we discuss related works. Section 3 details our experimental setup, methodology for capturing EM traces, and adversarial model. The experiments we performed and the results thereof are discussed in Section 4. The implications, limitations, and pointers for future work are then discussed in Section 5. Finally, our conclusions from this study are presented in Section 6.

2 RELATED WORK

One of the first works to mention the possibility of using side channels to reverse engineer operations performed by a device is the seminal work on Differential Power Analysis (DPA) presented by Kocher et al. [15]. Several works followed in subsequent years that put this idea into practice on low-end devices. Specifically, Eisenbarth et al. propose a methodology to obtain the program code running on a microcontroller by evaluating its power consumption [8]. They build 41 templates of individual instruction cycles and evaluate their approach on a PIC16F687, achieving a recognition rate of 70%. A more recent approach to template individual instructions was presented by Park et al [25]. Their methodology uses Kullback-Leibler divergence and PCA to extract features from a Wavelet transform of power measurements from an IoT device with an ATMega328P processor. These features are subsequently used to train a Machine Learning (ML)-based classifier.

In context of EM side channels, Balasch et al. propose an approach to identify operations performed by hardware trojans on an FPGA [3]. They achieve this by applying Welch's two-tailed T-test to distinguish between a compromised device and uncompromised (Golden) device. One of the first works targeting individual high-level operations on more complex CPUs is the work by Stone et al. [34]. They use an oscilloscope sampling at 1-10 GS/s to construct templates of individial instructions by aligning and averaging a set of 1,500 EM leakage signals from a MSP430F5529 device, and perform matched filtering using cross-correlation to classify them. Their methodology is able to distinguish 6 operand addressing modes at 93.22% accuracy, and performs slightly better than random guessing for distinguishing between different operations.

A recently published work that is conceptually most similar to ours is the work presented by Sayakkara et al. [31]. They propose an approach to identify cryptographic operations performed on a Raspberry Pi 3B+ and Arduino Leonardo using a HackRF Software Defined Radio (SDR). Their methodology utilizes a Multi-Layer Perceptron (MLP) classifier to classify 3 operations: DES, AES-128 and AES-256, and achieves over 82% classification accuracy. However, there are a number of important differences compared to our approach. First, they repeatedly perform these operations by encrypting a large file, and use 500 bins from a 200,000-point Fast Fourier Transform (FFT) as inputs to their MLP network. As such, their methodology can only detect operations that are performed continuously for an extended duration, rather than individual operations. Second, they do not consider the extraction of start and end times of individual operations from the EM trace. In our approach, both of these limitations are removed.

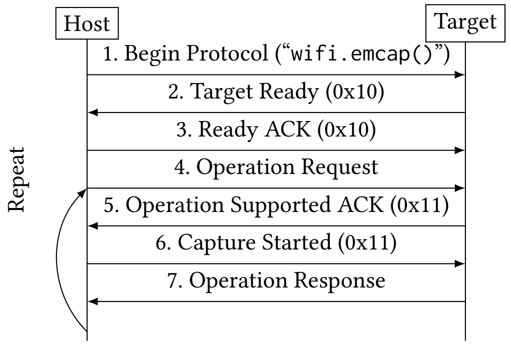


Figure 1: Overview of the serial-over-USB protocol performed between the host machine and NodeMCU to trigger the execution of operations.

Finally, in recent years, several works that use ML classifiers have been published in the domain of RF fingerprinting, where radio waves are used as features for the identification of devices or modulation schemes, rather than side-channel leakage [20, 24, 30].

3 METHODOLOGY

3.1 Experimental setup

The Device Under Test (DUT) used for the experiments discussed in this section is a single NodeMCU Amica device, which features the Espressif ESP8266EX Wi-Fi module. This module is widely used in IoT appliances such as smart light bulbs, meters, and sensors. The ESP8266EX features a Tensilica L106 32-bit RISC processor, which has a default clock speed of 80 MHz (overclockable to 160 MHz) [9]. To improve the SNR of leaked EM emissions, we removed the metal shielding cap of the device prior to analysis.

For capturing EM traces, we used a TekBox TBPS01 EM probe with a 40 dB low-noise amplifier connected to a Universal Software Radio Peripheral (USRP) B210 [10]. The USRP is connected over USB3 to a desktop workstation with an Intel Xeon CPU E5-1620 v2 @ 3.70GHz CPU and NVIDIA GeForce GTX 970 GPU. The probe is positioned closely above the crystal oscillator and IC of the NodeMCU, and is kept in place using adhesive tape.

3.2 Custom firmware and protocol

To make the acquistition of a clean dataset of EM traces more tractable, we developed custom firmware for the profiling device that allows it to interact with a host machine. This firmware is based on `nodemcu-firmware`, an open source Lua-based interactive firmware for ESP8266 and related chipsets [35]. The custom firmware implements a simple protocol that is performed over USB prior to each operation measurement. It allows the host to instruct which operation the device must perform, as well as which data must be provided as input. For example, in case of a cipher, the host can specify the key and plaintext to use. Figure 1 shows the messages exchanged during the protocol.

The host machine initiates the protocol and sends commands to the NodeMCU to perform random operations with random inputs. This executed operation can come from the following set: OpenSSL AES, native AES, TinyAES, OpenSSL DES, SHA1-PRF, HMAC-SHA1, SHA1, SHA1Transform, and "no operation". Of these operations, native AES, SHA1-PRF, HMAC-SHA1, SHA1, and SHA1-Transform are implemented by default in the ROM of the NodeMCU, whereas OpenSSL AES, TinyAES, and OpenSSL DES were added to the firmware of the device.

To mitigate effects related to caching, we warm up the cache by executing the operation once before the `Operation Supported ACK` message is sent, which is before the capture itself has started. This effect will be discussed in more detail in Section 5. After the warm-up and when the host machine receives the `Operation Supported ACK` packet, the USRP is tuned to a center frequency of 240 MHz, which corresponds to the third harmonic of the CPU clock, and capture is started. In our experimental setup, we empirically determined that this frequency was least influenced by transmissions from external radios. It is important to note that EM leakage produced by the processing of messages 6 and 7 will consequently be present in the measurements. A solution to this issue will be discussed in Section 4.6.

3.3 Capture and storage

For our experiments, we captured 3 datasets of EM emanations that were recorded during random-input executions of the operations listed above. To this end, we used the GNU Radio framework [13] with the USRP sampling at the maximum sample rate of 56 MS/s. Each dataset contains a separately recorded "test set", which is used for the evaluation of our experiments. The "random label" dataset contains 69,632 traces of random operations, totaling 60.9 GB. The "random label with bounding boxes" dataset is comprised of 768 manually labeled traces from the "random label" dataset. Finally, the "Wi-Fi connect" dataset consists of 3 traces of a complete Wi-Fi connection procedure from the NodeMCU to a `hostapd` AP, totaling 5.1 GB of data. The storage format and structure of the datasets is described in Appendix A.1.

3.4 Adversarial model

Recall from the introduction of this paper that the goal of the adversary is to capture EM emissions from the DUT and determine which operation out of a set of known operations is being performed at a particular point in time, based on a single trace. Hence, we assume the adversary is in possession of an identical device, which can be used to create EM templates for the operation of interest.

In our experiments, we further distinguish between two types of adversaries:

- **Grey-box adversary**: This adversary has a coarse notion of when operations of interest will take place on the DUT. They can trigger these operations on demand through a communication channel and measure EM emissions during this period. For example, the adversary could achieve this by sending a Wi-Fi frame or by modifying the firmware to externally trigger the unkown operation via GPIO or a serial communication channel. The goal of the adversary is then to determine the type of operation performed and its start and end times within the EM trace.
- **Black-box adversary**: This adversary has no control over the firmware or inputs of the DUT, i.e. operations cannot be

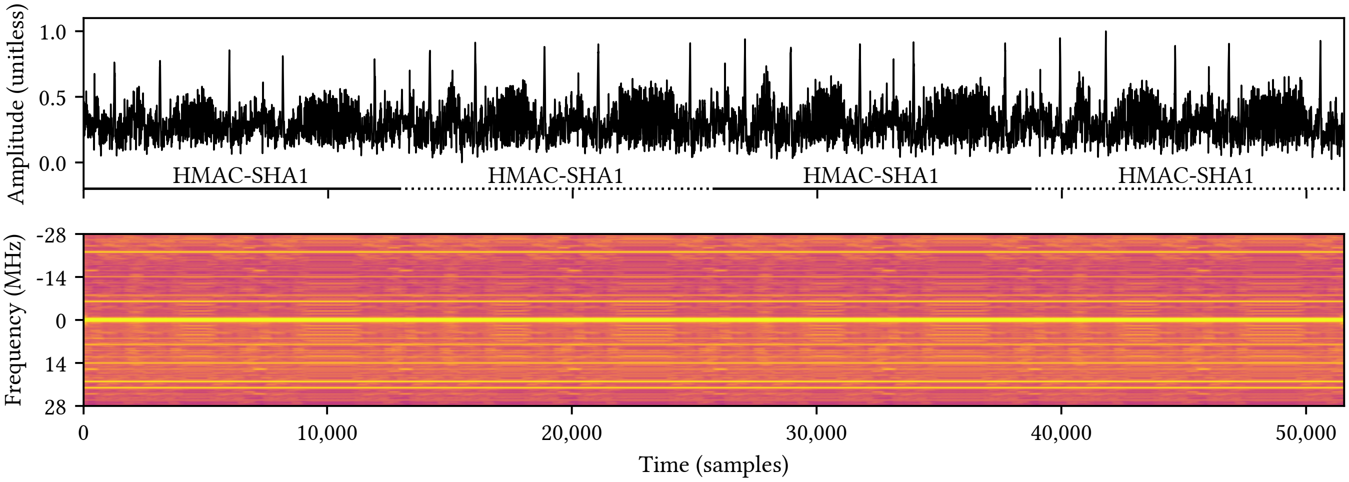


Figure 2: Example of a low-pass filtered amplitude-based (top) and STFT-based (bottom) template signal for SHA1-PRF, created by averaging EM emissions from a NodeMCU device that were measured using a USRP B210 sampling at 56 MS/s. Different stages of the algorithm, such as the 4 HMAC-SHA1 invocations, can be visually distinguished.

triggered on demand. They can only observe its EM emissions and must therefore be able to detect operations of interest in real time as they occur spontaneously.

3.5 Operation templates

For both ML and classical template matching, creating good templates requires solving two challenges: (i) sufficient measurements need to be made in order to be able to model the operation under different conditions and (ii) the measurements must be preprocessed (i.e. demodulated, filtered and normalized) in order to improve their quality. Furthermore, since the EM radiation is affected by various device-specific factors such as circuit geometry, coupling effects, and operation implementation [2, 21], templates are specific to the DUT, which justifies the requirement to have an identical device available for profiling.

In this work, we will consider two transformations of the complex EM trace as features for creating templates and classifying operations. The first is the AM-demodulated signal or instantaneous amplitude, defined as $|x(t)|$ with $x(t) \in \mathbb{C}$. We consider this feature since some types of EM leakage are known to manifest as Amplitude Modulation (AM) signals [2]. The instantaneous amplitude is furthermore commonly used to perform SCA in practice. A disadvantage of solely considering the instantaneous amplitude however, is that all frequency-related information of the EM trace is discarded. Since previous works have indicated that useful side-channel information may be found in the frequency domain as well [2, 12, 31, 36], we consider the STFT as a second feature in our experiments. The STFT is obtained by applying a Fourier transformation to segments of a given size w of a complex signal.

3.5.1 *Classical template matching amplitude templates.*

To create a template for classic amplitude-based template matching, we start by using the protocol described in Section 3.2 to capture a set $\mathcal{S}_o$ of n_o traces of length n_x for each operation $o \in O$. Then, the following steps are performed:

(1) For each operation, we visually locate a part of one trace where the operation is taking place, for example by looking at the signal envelope or waterfall plot. We will refer to this part of the trace as the "bootstrap signal" $b_o(t)$ of length n_b.
(2) Low frequency noise introduced by the chip is removed from the bootstrap signal and traces in $\mathcal{S}_o$ using a Butterworth high-pass filter, and the signal is AM demodulated. We then apply another high-pass filter to remove low-frequency noise from the signal envelope.
(3) The bootstrap signal $b_o(t)$ is chosen as a reference trace and correlated with the remainder of traces $x(t) \in \mathcal{S}_o$ using a sliding-window Zero-mean Normalized Cross-Correlation (ZNCC) denoted $c(t)$ as follows:

$$c_{b_o,x}(t) = \frac{\sum_{i=1}^{n_b}(b_o(i) - \overline{b_o})(x(i+t) - \overline{x}(t))}{\sqrt{\sum_{i=1}^{n_b}(b_o(i) - \overline{b_o})^2}\sqrt{\sum_{i=1}^{n_b}(x(i+t) - \overline{x}(t))^2}}$$

where $\overline{b_o}$ and $\overline{x}(t)$ indicate the mean of the bootstrap signal and windowed mean of the trace respectively. We use the $\arg\max$ of the correlation trace to determine the relative offset between the reference trace and other traces in $\mathcal{S}_o$.
(4) Finally, we average the aligned traces over the trace index axis to obtain the signal $z_o(t)$:

$$z_o(t) = \frac{1}{n_o}\sum_{i=1}^{n_o} x_i(t),\ x_i \in \mathcal{S}_o \in \mathbb{R}^{n_o \times n_x}$$

According to the law of large numbers and assuming the noise is Gaussian, a larger number of traces n_o will yield an average signal z_o closer to the expected value, i.e. a template that is free of noise. We will refer to such signals as "template signals".

Figure 2 (top) shows an example of an amplitude-based template signal for the SHA1-PRF algorithm, obtained by averaging 1,739 signals sampled at 56 MS/s. Note that we can visually distinguish patterns related to the execution, e.g., for SHA1-PRF we can identify the four iterations of HMAC-SHA1 taking place. Appendix B shows an overview of all template signals for the operations considered in this paper.

3.5.2 Classical template matching STFT templates. For the STFT-based templates, the following steps are performed:

(1) The bootstrap signal $b_o(t)$ for each operation is transformed using an STFT with FFT size of 512 and overlap of 90%. The FFT size can be tweaked to trade off time resolution versus frequency resolution. For SHA1Transform, which is the shortest operation in our set (21.43 µs), this size and overlap configuration results in approximately 23 FFT windows containing part of the operation.
(2) Analogous to steps 3 and 4 for the amplitude-based templates, we perform a sliding-window ZNCC with the bootstrap signal for each of the traces in the set, and average the resulting aligned STFT traces to obtain the template signal for each operation.

Figure 2 (bottom) shows an example of an STFT template signal for the SHA1-PRF algorithm, obtained by averaging 7,192 signals sampled at 56 MS/s. Again, we can visually distinguish changes in frequency-related features when the individual HMAC-SHA1 invocations happen during SHA1-PRF.

3.5.3 Machine-learning based templates. For creating ML templates we follow a similar approach, except we do not need to manually select a bootstrap signal and align the traces. We filter the traces in $\mathcal{S}_o$ and perform AM demodulation or a STFT transform analogous to the classical templates. Next, we normalize the trace set such that the dynamic range falls within the interval $[-1, 1]$ for amplitude-based features and $[0, 1]$ for STFT-based features. The resulting traces comprise the training set, which we will optimize with respect to an objective function to automatically learn the relationship between input features and their corresponding operation.

4 EXPERIMENTS

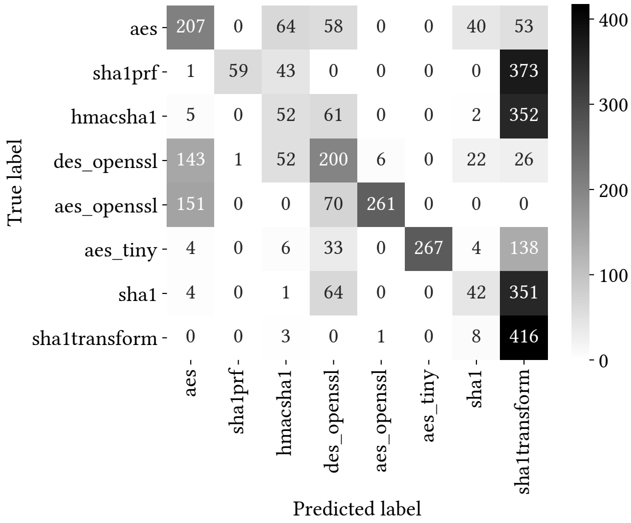
4.1 Classical template matching baseline

As a baseline for detecting operations of interest in a trace of EM emissions, we evaluate the "random label" dataset using two metrics that are commonly used in the domains of SCA and computer vision for template matching: ZNCC and Mean Squared Error (MSE). We will assume a grey-box adversary who has created templates for each of the operations of interest as described in Sections 3.5.1 and 3.5.2, and whose goal it is to distinguish which operation was performed in a given test trace, based on the measured EM leakage alone. Note that we assume the worst-case scenario, where the adversary has only one measurement of the operation of interest available. In practice, an adversary could (i) take multiple measurements of the same operation, synchronize them in time and take the average signal in order to improve the Signal-to-Noise Ratio (SNR) of the operation or (ii) perform majority voting on the individual measurements to determine the correct class.

4.1.1 Zero-mean normalized cross-correlation. When using the ZNCC, we correlate the EM trace with the template for each operation o and choose the operation with the highest correlation as the predicted class $\hat{o}$:

$$\hat{o} = \arg\max_{o \in O}(\max_t(c_{b_o,x}(t)))$$

Figure 3 shows the confusion matrix that is obtained when using this methodology to classify EM traces from the "random label" test dataset. The accuracy of using ZNCC to classify traces is 36.74%, with a macro-average precision and recall of respectively 47.23% and 37.38%. Notice that many of the incorrect predictions can be attributed to the SHA1Transform operation. The reason for these incorrect classifications is twofold. First, since this operation has a duration of only 21.43 µs (approximately 1,200 samples), its template has a higher probability of matching with random noise than other operations. Second, the ZNCC struggles to differentiate between operations that are contained within other operations. For example, SHA1Transform is called as a subprocedure in the SHA1, HMAC-SHA1, and SHA1-PRF operations. Consequently, we see a higher number of false positives for SHA1Transform when these operations take place.



	aes	sha1prf	hmacsha1	des_openssl	aes_openssl	aes_tiny	sha1	sha1transform
aes	207	0	64	58	0	0	40	53
sha1prf	1	59	43	0	0	0	0	373
hmacsha1	5	0	52	61	0	0	2	352
des_openssl	143	1	52	200	6	0	22	26
aes_openssl	151	0	0	70	261	0	0	0
aes_tiny	4	0	6	33	0	267	4	138
sha1	4	0	1	64	0	0	42	351
sha1transform	0	0	3	0	1	0	8	416

Figure 3: Confusion matrix for the classification of EM traces from the "random label" test set, using ZNCC.

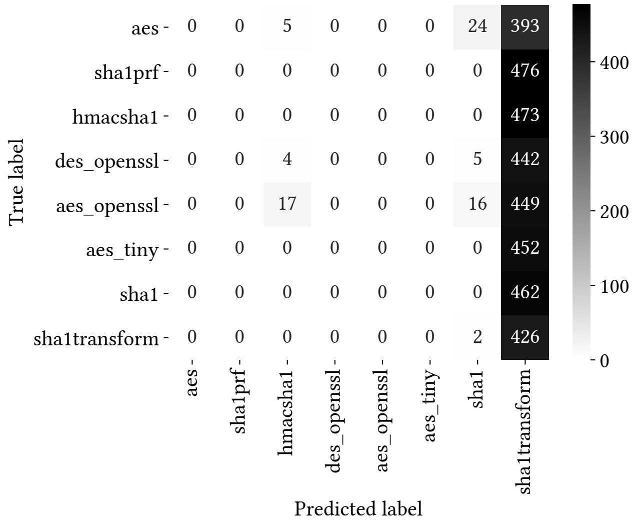
4.1.2 Mean squared error. For the MSE metric, we choose the operation with the lowest squared error as the predicted class $\hat{o}$:

$$d_{b_o,x}(t) = \frac{1}{n_b}\sum_{i=1}^{n_b}(b_o(i) - x(i+t))^2$$

$$\hat{o} = \arg\min_{o \in O}(\min_t(d_{b_o,x}(t)))$$

The confusion matrix when classifying the "random label" test dataset is shown in Figure 4. Note that the MSE is more sensitive to bias and scaling compared to the ZNCC, since we do not subtract the means of the signals or normalize by their standard deviations. In this case, the MSE almost always matches with the

SHA1Transform operation on a patch of noise, resulting in a macro-average precision of only 1.19%.



True label \ Predicted label	aes	sha1prf	hmacsha1	des_openssl	aes_openssl	aes_tiny	sha1	sha1transform
aes	0	0	5	0	0	0	24	393
sha1prf	0	0	0	0	0	0	0	476
hmacsha1	0	0	0	0	0	0	0	473
des_openssl	0	0	4	0	0	0	5	442
aes_openssl	0	0	17	0	0	0	16	449
aes_tiny	0	0	0	0	0	0	0	452
sha1	0	0	0	0	0	0	0	462
sha1transform	0	0	0	0	0	0	2	426

Figure 4: Confusion matrix for the classification of EM traces from the "random label" test set, using MSE.

4.1.3 STFT template matching. For our classical template matching experiment using the STFT templates described in Section 3.5.2, we used the ZNCC and MSE metrics to classify test traces from the "random label" dataset analogously to amplitude-based template matching. The results of this experiment are very similar to Figure 4: the templates frequently match with noise following the actual operation, resulting in many false positives for the OpenSSL AES class (when using MSE) and SHA1Transform class (when using ZNCC). The confusion matrices are provided in the supplimentary material of this paper (see Section A.2).

4.1.4 Discussion. Based on the previous experiments, we conclude that classical template matching is infeasible for classifying operations based on a single EM trace. The low SNR of operations of interest and the use of only local features in the trace causes the templates to frequently match with noise or overfit to one specific class.

We note that the accuracy of classical template matching can be greatly improved by either cropping the EM trace closer to the operation of interest itself or by aligning and averaging multiple test traces before performing the classification. Both of these approaches result in less noise being present in the trace. However, recall from Section 3.4 that we assume that an adversary has only a coarse notion of when the operation starts and ends, and that they must determine which operation is taking place based on a single EM trace. Hence, they cannot crop the trace to the operation of interest or create a less noisy signal by averaging multiple traces.

In the next sections, we will devise an approach to overcome this problem, allowing an adversary to identify and locate the operation of interest based in a single EM trace.

4.2 1D CNN classification

4.2.1 Architecture. Over the past years, a large variety of CNN architectures have been proposed and applied to solve challenges in the domains of image classification, object detection [29], speech recognition[37], side-channel analysis [27], and many others [18]. In convolutional layers of a neural network, kernels or "filters" are optimized with respect to a given loss function. When such a layer of filters is convolved with an input, it essentially acts as a feature detector for the next convolutional layer [17]. This is in contrast to fully connected layers, where each feature depends on all features of the previous layers.

In context of SCA, Benadjila et al. proposed a CNN design to extract features from EM traces of a side-channel protected AES implementation [27]. Their design was based on VGG-16, a CNN for image recognition introduced earlier by Simonyan et al [33]. The architecture they propose expects 700-point inputs of EM leakage measurements taken during the first round AES, which are passed through five blocks of convolutional / average pooling layers and three fully connected layers [27]. Similar convolution-based architectures were implemented in recent works to perform side-channel attacks on cryptographic algorithms [26, 38, 40].

For this experiment, we designed a CNN architecture, inspired by the architecture of Benadjila et al., to accomodate the needs of an adversary who operates under the assumptions of our grey-box adversarial model (see Section 3.4). Specifically, recall that we assume the adversary can trigger an operation through external means and measure the resulting EM emissions. For example, the adversary could trigger a series of AES decryption operations by sending a Wi-Fi frame to a CCM-mode enabled device.

The fixed 700-point input required by the architecture presented by Benadjila et al. translates to a maximum trace duration of 12.5 µs when sampling at 56 MS/s, which is too low for our purposes: the longest operation of interest we consider, SHA1-PRF, has an execution time of approximately 928 µs (51,968 samples) on the DUT. Furthermore, when the adversary triggers the operation, it does not start instantaneously due to various transmission and processing delays. With these delays in mind, a large enough window of samples must be considered such that the whole operation is contained within it. For the protocol described in Figure 1, we experimentally determined that on average, the delay between the start of the measurement and the start of the operation of interest is at least 357 µs (20,000 samples), whereas the total duration of the operation rarely exceeds 2.34 ms (131,072 samples). As such, the traces from the "random label" datasets described Section 3.3 were truncated to this interval, which allows us to use traces of a fixed 131,072-point duration as inputs to the CNN.

To make computations with large inputs more tractable, we reduce the number of filters per convolutional layer and perform average pooling with wider filters and strides. In addition, we also add a number of convolution and pooling layers in order to increase the receptive field of the CNN. Lastly, we define an additional "noise" class that represents the case where no operation besides the serial protocol itself (i.e., the trigger) is taking place, resulting in a total of 9 output neurons used as predictors for the operation classes. The final architecture is shown in Figure 5. The CNN was implemented using the Keras [6] ML framework with a

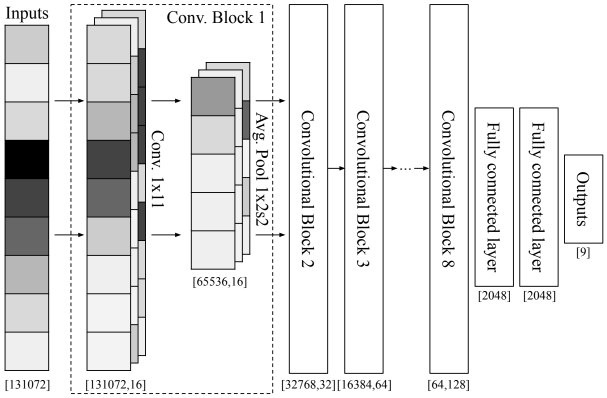


Figure 5: Overview of the CNN architecture used for classifying amplitude-based traces in a grey-box adversarial model.

Tensorflow backend [1]. The implementation has been made available publicly on Github[1].

4.2.2 Training and evaluation. We trained the architecture detailed in the previous section on the "random label" dataset training data for one epoch on the desktop workstation mentioned in Section 3.1 by optimizing the cross-entropy loss using the Adam optimizer [14]. Since training examples can theoretically be generated and automatically labeled ad infinitum, it should not be necessary in practice to perform data augmentation or to perform multiple iterations over the same training data. In addition, doing the latter may lead to overfitting. Before feeding the signals to the inputs of the CNN, the signals were preprocessed as described in Section 3.5.3.

The confusion matrix after evaluating on the test set is shown in Figure 6. Overall, we obtain a 94.43% accuracy, with a macro-average precision of 94.93% and macro-average recall of 94.06%. Compared to classical template matching, these results indicate that our CNN is much better at distinguishing operations that are contained within other operations, such as `SHA1Transform`.

4.3 2D CNN classification

4.3.1 Architecture. Similarly to the classical template matching experiments, we examined the effectiveness of utilizing the STFT as input features for a CNN. The architecture that we used is shown in Figure 7.

To prepare the inputs to the CNN, each trace is windowed or padded to a length of 131,328 samples and subsequently transformed using a 512-point STFT with 50% overlap. This results in a square 512 × 512 input vector per training example. The values of the inputs are normalized to the interval $[0, 1]$ before they are fed to the network. The input layer can be interpreted as a collection of images, which are passed through 5 2D convolutional blocks similarly to the VGG-16 architecture.

4.3.2 Training and evaluation. Identical to Section 4.2.2, we trained the 2D CNN on the "random label" dataset training data for one epoch on our desktop workstation. The confusion matrix after evaluating on the test set is shown in Figure 8.

[1] https://github.com/rpp0/em-operation-extraction

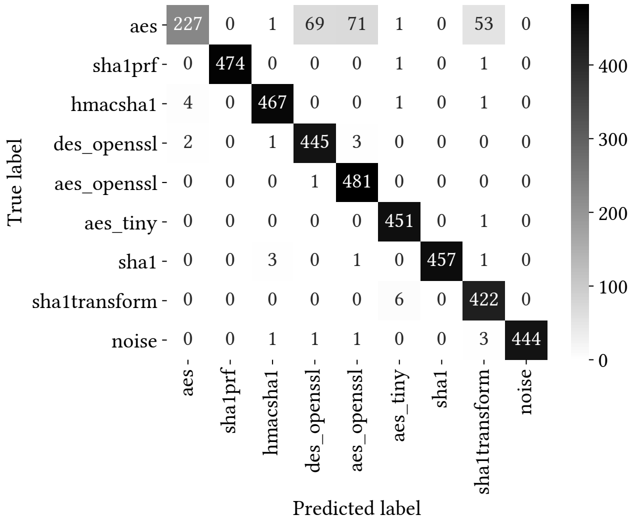


Figure 6: Confusion matrix for the classification of EM traces from the "random label" test set, using our 1D CNN.

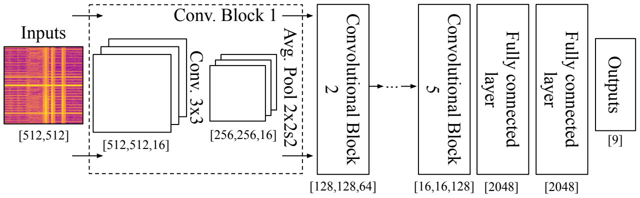


Figure 7: Overview of the CNN architecture used for classifying STFT-transformed traces in a grey-box adversarial model.

With an accuracy, precision and recall of respectively 94.38%, 95,92% and 94.01%, the results are comparable to the 1D CNN.

4.4 1D CNN classification with extraction

4.4.1 Architecture. While the CNN architectures described in the previous sections allow a grey-box adversary to determine which operation was performed within a time segment of fixed duration, they cannot determine where precisely in the time segment the operation of interest takes place. This information would be especially useful in context of SCA: if the CNN can automatically determine the start and end boundaries of an operation of interest, the adversary is no longer required to modify firmware of the DUT in order to incorporate triggers before or after an operation of interest.

To achieve this, we extend the output layer of the CNN to predict two additional outputs: (i) the midpoint between the start and end of the operation and (ii) the width or duration of the operation of interest. These quantities are sufficient to predict a bounding box around the operation while simultaneously predicting its class label. Conceptually, this approach to predict a bounding box is similar to the work presented by Redmon et al. [29], except we do not need to predict multiple bounding boxes or divide the trace

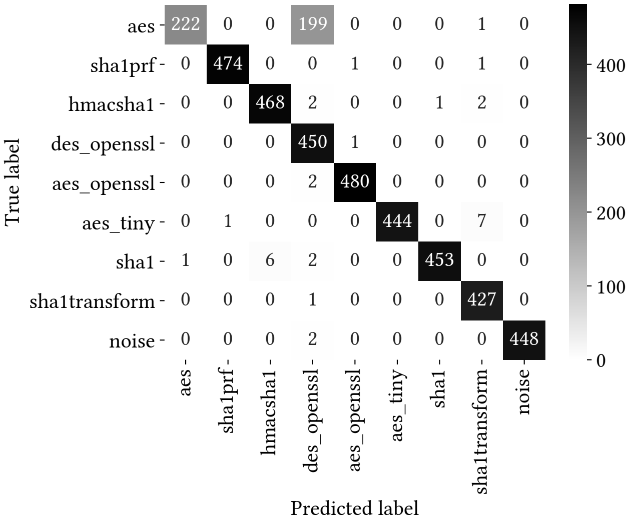


Figure 8: Confusion matrix for the classification of EM traces from the "random label" test set, using our 2D CNN.

into multiple segments because an adversary would not trigger the DUT to perform multiple operations within the same time frame. Both the midpoint and width are normalized with respect to the number of samples in a segment, such that their values lie within the interval $[0, 1]$.

4.4.2 Training and evaluation. For training the updated CNN, we can initialize the model parameters with the trained model parameters obtained from Section 4.2, and only retrain the last layer. We use the "random label with bounding boxes" dataset as the inputs. Recall from Section 3.3 that the operation bounding boxes for the training data in this dataset were manually labeled. Hence, we cannot generate training examples on the fly as before. To prevent the model from overfitting on this limited set of labeled bounding boxes, we perform data augmentation by rolling the trace and bounding box randomly along the time axis. Here, we limit the rolling operation such that the distance between the left bound and right bound is unchanged with respect to the original trace. This prevents any random splits inside the bounding box of the operation of interest. The data augmentation procedure ensures that bounding boxes and operations can occur at any position within the 131,072-point time segment. We use a custom loss function to optimize the classification and bounding box parameters simultaneously. Given a one-hot encoded class label y, bounding box width w, bounding box midpoint m and their respective predictions $\hat{y}$, $\hat{w}$, and $\hat{m}$, this loss function is defined as:

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{n_o} y_i \log(\hat{y}_i) + \lambda_b \mathbb{1}_b [(w - \hat{w})^2 + (m - \hat{m})^2]$$

where $\mathbb{1}_b$ is an indicator function that is one when a bounding box is defined for the training example and λ_b is a user-configurable parameter that can be used to tweak the importance of loss resulting from incorrect bounding box predictions. In our work, we set λ_b to 10^2. Note that the indicator function allows to ignore the bounding box loss altogether when the "no operation" class is predicted. Finally, the total cost of a single training step is the average loss of the training batch $B \subset \mathcal{S}$ containing n_B training examples.:

$$C(y, \hat{y}) = \frac{1}{n_B} \sum_{i=1}^{n_B} \mathcal{L}(y^{(i)}, \hat{y}^{(i)})$$

The confusion matrix after training for approximately one day on the "random label with bounding boxes" training set and evaluating on the test set is shown in Figure 9.

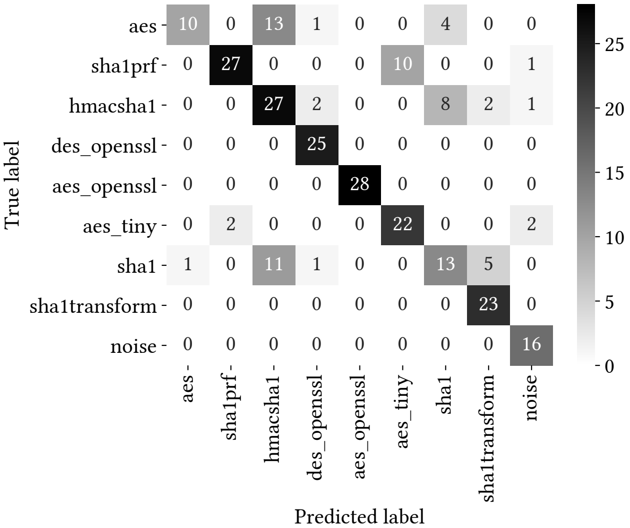


Figure 9: Confusion matrix for the classification of EM traces from the "random label with bounding boxes" test set, using our 1D CNN.

For this experiment, we obtain an accuracy of 74.90%, and a precision and recall of respectively 77.84% and 77.87%. The mean absolute error of the bounding box predictions with respect to the ground truth for 239 traces containing an operation is 429.81 μs (24,069 samples) per trace. Although the classification accuracy is lower compared to the experiments without bounding boxes, we still believe this is a good result given the limited set of 256 labeled training examples.

4.5 2D CNN classification with extraction

Analogous to the 1D CNN scenario from the previous section, we modified the 2D architecture from Section 4.3 to predict the midpoint and width of the operation of interest. However, note that since we applied a 512-point STFT transformation with 50% overlap to the traces, the granularity of predicted start and end boundaries of the operation of interest is limited to units of 4.57 μs (256 samples). The same data augmentation technique from Section 4.4 was applied during training. Figure 10 shows the confusion matrix after training for approximately one day on the "random label with bounding boxes" training set and evaluating on the test set. This methodology achieves an accuracy, precision, and recall of respectively 96.47%, 96.69%, and 96.78%. The mean absolute error of the bounding box predictions with respect to the ground truth for 239 traces is 34 µs (1,904.02 samples) per trace.

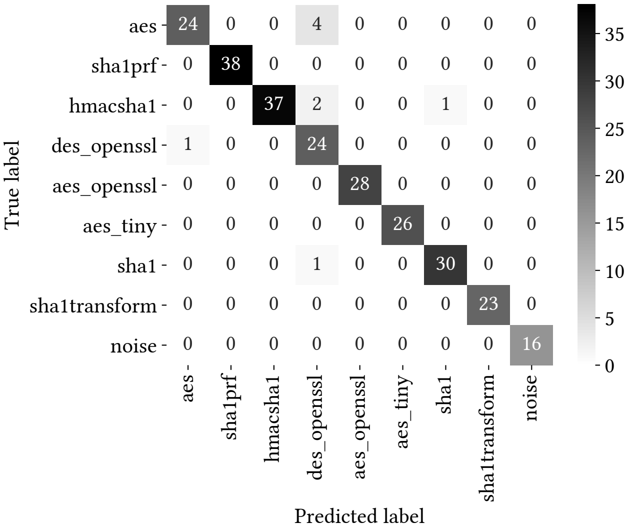


Figure 10: Confusion matrix for the classification of EM traces from the "random label with bounding boxes" test set, using our 2D CNN.

4.6 Detecting operations in a Wi-Fi connect

4.6.1 Improved data augmentation. Up until now, we assumed a grey-box adversary who can trigger operations on demand from the DUT. In a black-box adverserial model (see Section 3.4), the adversary cannot perform any interactions with the DUT. This means that they must continously capture EM emissions and distinguish operations of interest on the fly. For this experiment, we will therefore use the "Wi-Fi connect dataset", which contains 3 full traces of the NodeMCU performing the `wifi.sta.connect()` function to connect to an AP.

Considering our CNN is trained to distinguish the "noise" class from operations as evidenced in Section 4.4.2, one could segment the trace into overlapping sections of length 131,072, and feed them individually to the network. However, in contrast to the previous experiments, for the "Wi-Fi connect dataset" we do not know which segments contain operations since they occur spontaneously, rather than being triggered externally. This prevents us from evaluating the accuracy of our models. To resolve this issue, we insert the segments containing operations of interest from the "random label with bounding boxes" test set inbetween random snippets from a "Wi-Fi connect dataset" trace. This gives us the benefit of being able to simulate a black-box environment while retaining the ability to evaluate the accuracy of our model.

Finally, since the noise profile measured during the execution of the protocol from Figure 1 substantially differs from the noise profile measured during a real Wi-Fi connection, we need to train our model to be able to recognize these different types of noise. We achieved this by modifying our augmentation procedure from the previous experiments to insert the operation of interest inbetween a random snippet from a "Wi-Fi connect dataset" trace with probability 1/2.

4.6.2 Evaluation. To evaluate our models, we insert operations from the "random label with bounding boxes" test set into a random snippet from the "Wi-Fi connect dataset" trace with probability 1. For the 1D CNN, the accuracy, precision and recall decrease to respectively 38.04%, 63.84%, and 40.44%. Similarly, the 2D CNN performance decreases to an accuracy, precision and recall of respectively 55.29%, 78.67%, and 54.36%. Visual inspection of incorrect classifications reveals that they are often caused by microarchitectural effects of the NodeMCU. We will discuss this problem in greater detail in Section 5 and consider improving the accuracy in black-box scenarios as an interesting challenge for future work. The confusion matrices for this experiment can be found in Section A.2.

As a final experiment, we evaluated the models trained with the improved data augmentation method on the "random label with bounding box" test set, and found that these models perform slightly better for the 1D CNN and similarly for the 2D CNN on the "random label with bounding box" test dataset classification: the accuracy for the 1D CNN increases to 81.96%, whereas the accuracy for the 2D CNN becomes 96.86%.

5 DISCUSSION

While the results from our experiments indicate that the use of CNNs can be beneficial for detecting operations in context of SCA and reverse engineering, we believe there are still many challenges left to be solved. In this section, we will provide a detailed discussion of these challenges and how they relate to the domain of computer vision. Furthermore, we will discuss some of the limitations of this study, and provide pointers for future work.

5.1 Occlusion, lighting and depth

Assuming only one CPU is targeted when classifying operations, it should not be possible for two operations to take place at the same time. At first sight, this seems a major benefit compared to object detection in video or images, where objects of interest may appear under different lighting conditions, at different depths, or be occluded by other objects. However, for operation detection, "lighting and occlusion" in EM traces comes from microarchitectural effects that are hard to predict if the previous states of the device are unknown. For example, speculative execution, caching, pipelining and branch prediction all have an influence on the duration and SNR of operations of interest.

In particular for our experiments, we found that the OpenSSL implementations of AES and DES were the most difficult to localize in EM traces, even to the human eye. Figure 11 shows two EM traces of OpenSSL DES from our training set. Notice that the leftmost figure shows a large spike of EM emissions, occluding most of the operation. We postulate that this effect is caused by cache misses for two reasons. Firstly, we observe a similar effect for other operations if we do not perform a "warm-up" of the operation, meaning we execute the operation once before measuring its leakage. Secondly, the effect occurs mostly for the OpenSSL variants of the operations, even when performing a warm-up. These operations are the most memory intensive.

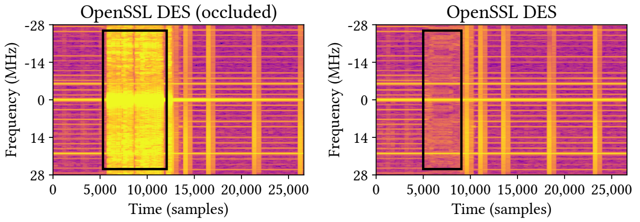


Figure 11: Example of occlusion caused by microarchitectural effects of the NodeMCU. The black bounding box indicates the interval where OpenSSL DES is performed by the device.

A limitation of our work is that we do not study the impact of microarchitectural effects such as caching. Rather, we implicitly assume the adversary can invoke the operation multiple times to mitigate caching effects to some degree and incorporate these effects as part of our classification model. Nevertheless, it would be useful to separately model these effects in future work, in order to improve the classification accuracy.

Finally, EM side-channel defenses such as masking and hiding could be considered as another form of occlusion, albeit deliberate occlusion by the programmer. Though such measures are designed to hide the data processed by the operation (e.g. the secret key of AES) rather than the overall appearance of the operation in an EM trace, they may still make it more difficult for an adversary to create operation templates. The effectiveness of such measures in this regard would be another interesting topic for future research.

5.2 Hardware and test setup

In this paper, we focused on detecting operations using commercial off-the-shelf SDR hardware such as the USRP B210 with only a single probe placed close to the CPU of the NodeMCU. A more powerful adversary might be able to install multiple probes at multiple physical locations near the DUT and use more expensive hardware in order to capture more features at a higher sampling rate. Using hardware with a higher sampling rate would allow to fingerprint operations from devices with a higher clock rate. Furthermore, devices under test with mixed-signal ICs may leak information over several meters, as indicated by Camurati et al. [5]. Therefore, in a future work it could be interesting to study the classification accuracy under these more powerful adversarial models.

Other factors in a test setup that can influence EM measurements are environmental effects such as the ambient temperature [25] or interference from other devices. Although we did not explicitly study the impact of these effects, the datasets presented in this work were recorded in an office environment over an extended period of time. As a result, we found that our trained models still perform well on EM traces captured at later times: for traces captured 24 days after the training set, we achieved similar accuracy. This experiment can be reproduced in a grey-box adversarial setting using the `realtime_capture.py` script provided in the Github repository, to classify traces in real time.

5.3 Bottom-up operation detection

The templates of the operations of interest in this paper were constructed using a top-down approach. That is, all subprocedures executed by the operation are considered to be part of the template. Another possibility would be to make templates for the subprocedures themselves, and infer which higher level operations take place based on observations of the subprocedures. As a concrete example, once could make a template only for `HMAC-SHA1`, and infer that `SHA1-PRF` was performed on the device by observing a sequence of 4 `HMAC-SHA1` invocations.

If leakage can be measured with sufficient resolution, it is possible to create templates of instruction-level events. Reconstructing higher-level operations from this information is an active area of research. Examples are the works of Park et al. [25] and Eisenbarth et al. [8]. However, these works focus on either low-end CPUs, such as the ATMega328P, or FPGAs. In both cases, there is a more clear relationship between EM leakage and executed operation. It would be interesting to investigate new methods for identifying instruction-level events in complex CPUs, such as the Tensilica L106 or CPUs present in regular desktops and laptops.

5.4 Generalization

While this study focused on identifying 8 operations from the NodeMCU, our methodology could be extended in future works to identify a larger variety of cryptographic operations or to identify operations over multiple hardware platforms. As a result, a more general model could be developed. Achieving this goal requires a number of challenging problems to be solved: on certain hardware platforms, operations might have a non-constant execution time over different iterations (e.g. due to optimizations), have a shorter or longer overall duration, or have different leakage properties due to differences in hardware architecture.

6 CONCLUSIONS

In this paper, we studied practical methods to classify and extract 8 possible operations of interest from singular EM emissions originating from a NodeMCU Amica device, using a USRP B210 SDR. We evaluated and compared classical template matching to classification using CNNs for two types of features from EM traces: the instantaneous amplitude and STFT. These methods were evaluated under both a grey-box and black-box adversarial model, where our best CNN models achieve an accuracy of respectively 96.47% and 55.29%. Lastly, we demonstrate how these models can be used to simultaneously predict the start and end times of operations within 34 µs of the ground truth on average.

ACKNOWLEDGMENTS

This research was partially funded by a Ph.D. Grant of the Research Foundation Flanders (FWO), grant number 1S14916N, and BOF. The authors wish to thank each of the anonymous reviewers for their insightful suggestions for improving this paper. We would also like to thank Tom Haber, Balazs Nemeth, Bram Vanherle, Tien Dang Vo-Huu and Marinos Vomvas for their valuable input, advice, and for the interesting discussions.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. 2003. The EM Side-Channel(s). In *Cryptographic Hardware and Embedded Systems*, Burton S. Kaliski, çetin K. Koç, and Christof Paar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 29–45.

[3] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. 2015. Electromagnetic Circuit Fingerprints for Hardware Trojan Detection. In *IEEE International Symposium on Electromagnetic Compatibility (EMC)*. IEEE, 246–251.

[4] Julien Brouchier, Tom Kean, Carol Marsh, and David Naccache. 2009. Temperature Attacks. *IEEE Security & Privacy* 7, 2 (2009), 79–82.

[5] Giovanni Camurati, Sebastian Poeplau, Marius Muench, Tom Hayes, and Aurélien Francillon. 2018. Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 163–177.

[6] François Chollet et al. 2015. Keras. https://keras.io.

[7] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. 2008. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In *Advances in Cryptology – CRYPTO*, David Wagner (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 203–220.

[8] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. 2010. Building a Side Channel Based Disassembler. In *Transactions on computational science X*. Springer, 78–99.

[9] Espressif. [n.d.]. *ESP8266EX – Low-Power, Highly-Integrated Wi-Fi Solution*. Retrieved February 21, 2020 from https://www.espressif.com/en/products/hardware/esp8266ex/overview

[10] Ettus Research. [n.d.]. *USRP B200/B210 Product Overview*. Retrieved February 21, 2020 from https://www.ettus.com/wp-content/uploads/2019/01/b200-b210_spec_sheet.pdf

[11] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. 2001. Electromagnetic Analysis: Concrete Results. In *Cryptographic Hardware and Embedded Systems*, Çetin K. Koç, David Naccache, and Christof Paar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 251–261.

[12] Daniel Genkin, Itamar Pipman, and Eran Tromer. 2015. Get Your Hands Off My Laptop: Physical Side-Channel Key-Extraction Attacks on PCs. *Journal of Cryptographic Engineering* 5, 2 (01 Jun 2015), 95–112. https://doi.org/10.1007/s13389-015-0100-7

[13] Ben Hilburn et al. 2020. *GNU Radio – The Free & Open Source Radio Ecosystem*. GNU Radio project. https://www.gnuradio.org/

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Annual International Cryptology Conference*. Springer, 388–397.

[16] Paul C Kocher. 1996. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Annual International Cryptology Conference*. Springer, 104–113.

[17] Yann LeCun and Yoshua Bengio. 1995. Convolutional Networks for Images, Speech, and Time Series. *The Handbook of Brain Theory and Neural Networks* (1995), 255–258.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. *Nature* 521, 7553 (2015), 436.

[19] Gaëtan Leurent and Thomas Peyrin. 2020. SHA-1 is a Shambles.

[20] Kevin Merchant, Shauna Revay, George Stantchev, and Bryan Nousain. 2018. Deep Learning for RF Device Fingerprinting in Cognitive Communication Networks. *IEEE Journal of Selected Topics in Signal Processing* 12, 1 (2018), 160–167.

[21] Olivier Meynard, Denis Réal, Sylvain Guilley, Florent Flament, Jean-Luc Danger, and Frédéric Valette. 2010. Characterization of the Electromagnetic Side Channel in Frequency Domain. In *International Conference on Information Security and Cryptology*. Springer, 471–486.

[22] David P Montminy, Rusty O Baldwin, Michael A Temple, and Mark E Oxley. 2013. Differential Electromagnetic Attacks on a 32-bit Microprocessor using Software Defined Radios. *IEEE Transactions on Information Forensics and Security* 8, 12 (2013), 2101–2114.

[23] Colin O'Flynn and Zhizhang David Chen. 2015. Side Channel Power Analysis of an AES-256 Bootloader. In *28th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 750–755.

[24] Timothy J O'Shea, Johnathan Corgan, and T Charles Clancy. 2016. Convolutional Radio Modulation Recognition Networks. In *International Conference on Engineering Applications of Neural Networks*. Springer, 213–226.

[25] Jungmin Park, Fahim Rahman, Apostol Vassilev, Domenic Forte, and Mark Tehranipoor. 2019. Leveraging Side-Channel Information for Disassembly and Security. *J. Emerg. Technol. Comput. Syst.* 16, 1, Article Article 6 (Dec 2019), 21 pages. https://doi.org/10.1145/3359621

[26] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. 2018. On the Performance of Convolutional Neural Networks for Side-Channel Analysis. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 157–176.

[27] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. 2018. Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. *IACR Cryptology ePrint Archive* (2018), 53. http://eprint.iacr.org/2018/053

[28] Jean-Jacques Quisquater and David Samyde. 2001. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security (E-SMART '01)*. Springer-Verlag, London, UK, UK, 200–210. http://dl.acm.org/citation.cfm?id=646803.705980

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 779–788.

[30] Shamnaz Riyaz, Kunal Sankhe, Stratis Ioannidis, and Kaushik Chowdhury. 2018. Deep Learning Convolutional Neural Networks for Radio Identification. *IEEE Communications Magazine* 56, 9 (2018), 146–152.

[31] Asanka Sayakkara, Nhien-An Le-Khac, and Mark Scanlon. 2019. Leveraging Electromagnetic Side-Channel Analysis for the Investigation of IoT Devices. *Digital Investigation* 29 (2019), S94–S103.

[32] Adi Shamir and Eran Tromer. 2004. Acoustic Cryptanalysis. *Presentation available from http://www.wisdom.weizmann.ac.il/~tromer* (2004).

[33] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).

[34] Barron D Stone and Samuel J Stone. 2015. Radio Frequency Based Reverse Engineering of Microcontroller Program Execution. In *National Aerospace and Electronics Conference (NAECON)*. IEEE, 159–164.

[35] The NodeMCU firmware contributors. 2020. *NodeMCU-firmware – Lua-based Interactive Firmware for ESP8266, ESP8285 and ESP32*. NodeMCU. https://github.com/nodemcu/nodemcu-firmware

[36] C Tiu. 2005. *A New Frequency-Based Side Channel Attack for Embedded Systems*. Master's thesis. University of Waterloo.

[37] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. In *9th ISCA Speech Synthesis Workshop*. 125–125.

[38] Guang Yang, Huizhong Li, Jingdian Ming, and Yongbin Zhou. 2018. Convolutional Neural Network Based Side-Channel Attacks in Time-Frequency Representations. In *International Conference on Smart Card Research and Advanced Applications*. Springer, 1–17.

[39] YongBin Zhou and DengGuo Feng. 2005. Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. *IACR Cryptology ePrint Archive* (2005), 388.

[40] Yuanyuan Zhou and François-Xavier Standaert. 2019. Deep Learning Mitigates But Does Not Annihilate the Need of Aligned Traces and a Generalized ResNet Model for Side-Channel Attacks. *Journal of Cryptographic Engineering* (2019), 1–11.

A ONLINE RESOURCES

A.1 Datasets

A.1.1 Structure. Batches of 16 traces are stored in separate files, structured according to the "ChipWhisperer" format as follows:

- ***_traces.npy**: Numpy array of data type `complex64` containing the EM traces.
- ***_textin.npy**: Numpy array of `uint8` lists containing the plaintext given as input to the operations.
- ***_knownkey.npy**: Numpy array of `uint8` lists containing the key used in the operations (if applicable).
- ***_meta.p**: Pickled metadata of dictionary containing the operation performed ("op") and operation bounding box ("`bound_left`" and "`bound_right`").

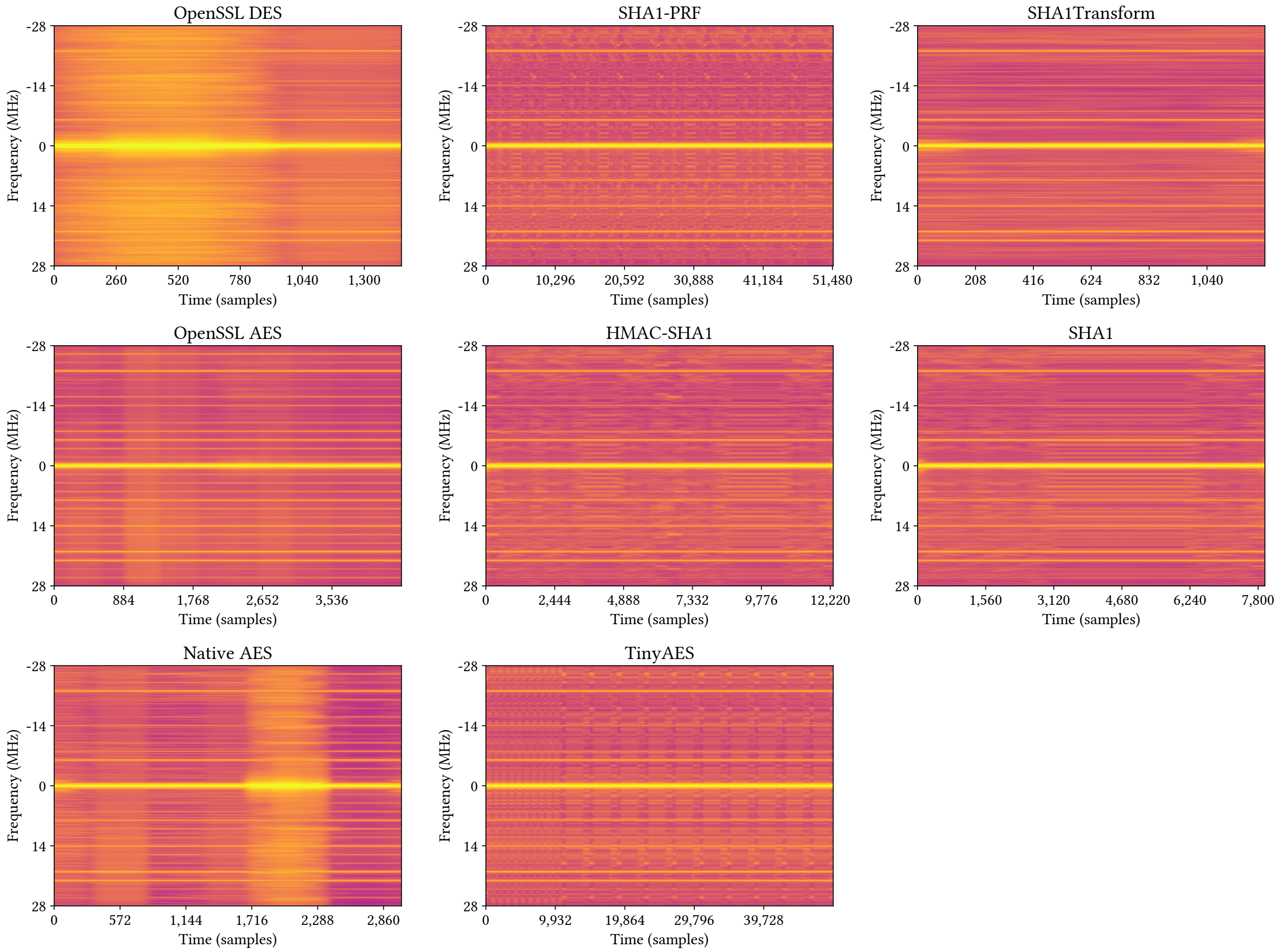


Figure 12: Overview of all STFT-based templates for the operations of interest considered in this paper.

A.1.2 Random label example counts.

- **nodemcu-random-train**: des_openssl: 7377, sha1prf: 7192, aes: 7339, aes_tiny: 7322, sha1: 7271, sha1transform: 7102, aes_openssl: 7284, noise: 7349, hmacsha1: 7300.
- **nodemcu-random-test**: sha1transform: 428, aes_openssl: 482, hmacsha1: 473, aes: 422, sha1prf: 476, des_openssl: 451, sha1: 462, aes_tiny: 452, noise: 450

A.1.3 Random label with bounding boxes example counts.

- **nodemcu-random-label-train**: sha1: 24, sha1transform: 30, noise: 36, des_openssl: 30, sha1prf: 25, aes_tiny: 33, hmacsha1: 32, aes_openssl: 25, aes: 21
- **nodemcu-random-label-test**: sha1prf: 38, sha1: 31, hmacsha1: 40, aes_openssl: 28, noise: 16, des_openssl: 25, aes_tiny: 27, aes: 28, sha1transform: 23

A.2 Trained models and results

The presented trained models and results are available at the WiSec dataset page http://wisecdata.ccs.neu.edu/ and on https://github.com/rpp0/em-operation-extraction.

B TEMPLATE SIGNALS

An overview of all STFT-based template signals is given in Figure 12.