Deep Learning based Side Channel Attacks on Intel CPU

Guilherme Rodrigues Lourenço

Academia Militar e Instituto Superior Técnico, Lisboa, Portugal

November 2022

Abstract

Nowadays, cryptographic implementations are increasingly exposed to the so-called side-channel attacks, which observe the leakage (power, time, etc.) produced by the cryptographic implementation to discover internal secrets. In this work, we aim to investigate the possibility and capability of a side-channel attacks using the power measured by the processor to acquire the encryption keys used by that device. To achieve this, we are going to introduce a software-based power side-channel attacks into an Intel processor with the objective of conquering undeserved access to the Intel Running Average Power Limit interface that shows values directly associated with the power consumption of the device, and then we will apply Deep Learning techniques using different Deep Neural Network architectures in order to be able to discover the whole key. For this Thesis, the data coming from the software measurements was acquired from simpler versions of the Advance Encryption Standard and then provided to the Neural Networks for training and implementation of the attack. The results obtained by the originated Neural Network models, when put under test, showed that leakage occurs and that along the same conditions of the data used to create the models, it is possible to discover the targeted secret byte.
Keywords:cryptographic implementations, side-channel attacks, security, deep learning, deep neural networks

1. Introduction

The world is in the middle of a transition phase, where all type of documents and information from the less relevant to the most critical, in the vast majority, are kept online. That is why security has been an increasing and significant concern in computing and communications systems. Cryptographic algorithms, including public-key ciphers, symmetric ciphers, and hash functions, create a set of characteristics that make it possible to construct security mechanisms that target specific objectives [20]. However, it should be assumed that attackers will not directly get through the computational complexity and break the cryptographic primitives employed in security mechanisms [19]. This type of attack is called a Side-Channel Attack (SCA), and it works because there is a correlation between the physical measurements taken during computations, such as power consumption, electromagnetic radiation, and computing time and the internal state of the processing device, which is itself related to the secret key that in several cases happen to be out of the security systems control [7]. Recently Machine Learning (ML) has become one of the most versatile and efficient new technologies for processing large amounts of data and finding underlying patterns that are difficult to recognize using other methods. Several studies [12] have shown that Machine Learning techniques, and more particularly, Deep Learning (DL) techniques have also been applied to time series data classification with outstanding results, and more specifically, these techniques have also been experimentally applied to SCAs with promising results [4]. For this reason DL is a subject of interest in this work, and it is speculated that employing DL classification techniques to perform a SCA on data that has been acquired solely using software may have a positive impact.

2. Background

2.1. Cryptography

Cryptography is the area of Information Security that protects sensitive data from unauthorized users. The kind of data covered by cryptography goes from resting data files to messages, passwords, and so many others. The secure transmission of information through insecure channels is allowed to ensure confidentiality. Moreover, a cipher and a secret key are required to achieve these goals. A cipher is a cryptographic algorithm responsible for performing encryption or decryption.

- **Encryption** is the process that converts data that usually is in a readable form, called plaintext, into an unintelligible form, called ciphertext;

- **Decryption** works oppositely; it transforms the ciphertext into the original plaintext form. The secrets represent the keys used by the cipher's encryption and decryption processes and are composed of large numbers.

There are several cryptographic services relevant to this work, that are guaranteed by two key types: symmetric and asymmetric.

2.1.1. Symmetric Cryptography

At least two communicating parties share symmetric keys. Symmetric keys are smaller, and the operations are faster than with asymmetric keys. The same key is used to encrypt and decrypt. The main disadvantage of this system is that sharing a secret key between different parties is usually not simple, especially thinking in a realistic way, where there are no secure channels to communicate.

Advanced Encryption Standard

One of the most popular symmetric-key algorithms is the Advanced Encryption Standard (AES). This algorithm uses 128-bit blocks for block cipher modes, and the keys can be 128, 192, or 256 bits. The algorithm performs 10, 12, or 14 transformation rounds, depending on the size of the cipher key used. Those transformations are usually applied on 16 bytes of data, called the state, represented as a 4x4 byte "state matrix". A round is composed of several operations that should transform the input into an output based on those transformations. Each round key is derived from the cipher key, using a key expansion procedure. Every round uses the same number of bits for the key since the beginning, and each round requires a different 128-bit key, referred to as "round key". During a round, four types of operations may be performed:

- **AddRoundKey:** In this operation, each byte from the state is combined with a byte from the round key using an XOR operation;
- **SubBytes:** In this operation, each byte from the state is replaced (in a non-linear fashion) by another byte, according to a lookup table. The process of replacing each byte by resorting to this lookup table is known as S-Box;
- **ShiftRows:** This operation applies a left shift to each state matrix row, except for the first one. The shifting amount applied to each row follows the formula $n-1$, where n is the row index, e.g., for the second row, all bytes are shifted one position to the left;
- **MixColumns:** In this operation, each column from the state matrix is linearly mixed, i.e., all the bytes from a column are combined to obtain a new column.

The last round must be the only that it is not wholly equal to the others, since the last operation is not performed.

2.2. Side-Channel Attacks

In an SCA, the attacker exploits the leakages of the system, intending to gather information about the executed operations. If enough information is collected, their later analysis will reveal the secret key.

The origin of these attacks dates back to 1996, when Kocher [11] broke RSA and Diffie-Hellman systems (asymmetrical encryption algorithms) based on the time it took to perform the operations. Three years later, as described in [10], he used power measurements to break the Data Encryption Standard (DES), the most commonly used symmetrical encryption algorithm at the time and proved capable of doing so on other algorithms. Given the importance of this family of attacks, under which most cryptographic algorithms have failed to resist when not adequately protected, a big concern is given to the corresponding countermeasures. These can range from software changes to extensive hardware modifications.

2.3. Computer Architecture

There are specific components in a computer that are essential to its functions, namely the Central Processing Unit (CPU). The CPU holds a limited number of storage locations, known as registers, a Control Unit (CU), and an Arithmetic Logic Unit (ALU). The clock is responsible for synchronizing the internal operations of the CPU with other components in the system. It is where calculations and logic operations occur.The CU organizes the ordering of steps responsible for doing system instructions. The ALU performs arithmetic operations such as addition and subtraction and logic operations [18]. Each Intel processor also comes with a new feature, the Running Average Power Limit (RAPL), which has a specific domain, the Power Plane 0 (PP0) that allows not only to measure the power consumed but also their performance [3].

2.4. Deep Learning: Basics, Classification and Review

Deep learning is a subdivision area of machine learning that uses tools and algorithms based on artificial neural networks, also known as neural networks (NN). These NNs are inspired by biological brains, and the neurons that compose them. Each NN is made up of neurons, or nodes, which conduct signals with information to each other, just like biological neurons transmit information through synapses.The nodes all work together to precisely recognize, classify, and describe objects within the data. Neural networks are effective because they are able to perform complex tasks very efficiently, notably by learning from a set of examples. Neural networks are created based on many neurons, that are all linked generating a connected graph organized in layers. These layers represent groups of neurons usually at the same depth in the network and operating in parallel, and they may be of different types. The most common ones are:

- Fully connected layer that happens when each neuron of one layer connects to all the neurons of the next layer;
- Convolutional layer especially useful in extracting structural redundancy within their input;
- Pooling layer usually appears after convolutional layers with the objective of not only reducing the data volume but also the number of weights in the next layers.

Based on the type of graph created by all the links, an NN may be classified as: 1) a Feed-Forward Neural Network (FFNN); or 2) a Recurrent Neural Network (RNN). FFNNs have inputs that go from the input layer to the output layer, without any cycles, which means that the output from one layer is only capable of affecting a future layer and never previous ones. Layers are formed by gathering many neurons and combining them, if many layers are combined, then neural networks are formed.

RNNs have cycles in the network, which allows the output from one layer to not only influence the next layers but also influence the same layer itself, or even the layers before. This type of NN, is very useful with inputs that have temporal correlation. Given that it can combine past information with current information to define the neuron output.

2.4.1. Deep Learning Network Classes

2.4.2. Convolutional Neural Network

Convolutional neural network (CNN) is a neural network architecture composed by five different types of layers: input, convolution, pooling, fully connected and output layers in order to produce a hierarchical decomposition of the input signal.

2.4.3. Recurrent Neural Network

Recurrent neural network (RNN) is a neural network architecture typically better at handling data evolving in time, like sequential or time series data. In an RNN, past information is able to influence current outcomes by taking outputs resulting from prior inputs and inserting them back into a network layer where current inputs are being processed. Overall, an RNN propagates the input signal forward as a regular network, but signals are also propagated back to the same neuron that produced it or to a previous one with the objective of keeping a context of what was previously processed. RNNs are able to model functions which have a dependency in time between the input and the output, thus creating cycles which give the network the capability to model dynamic temporal behaviors.

2.4.4. Convolutional Long Short-term Memory Network

In case of having data collected during successive periods of time, it is usual to characterize them as a time series, and the approach of using ConvLSTM layers is more than acceptable. It is a combination of RNN explained in Section 2.4.3 with CNN explained in Section 2.4.2 networks. It is a model based on LSTM, where the input layers are a mixture of convolutinal layers with time, offering the filtering capacity of the CNNs and the long-term memory of the LSTMs .

3. State of The Art

3.1. Power Analysis

Power Analysis (PA) is one of the most widely known and used branches of side-channel attacks nowadays. Of all types of SCAs known today, the literature on power analysis attacks and the relevant countermeasures is the largest. Power analysis attack is the current main research focus of side-channel attacks. This attack was first proposed by Kocher, Jaffe, and Jun in 1999 [10], where the power consumption was measured with an oscilloscope and used to completely get over the Data Encryption Standard (DES) algorithm. Power traces often leak meaningful amounts of information about the processed cryptographic key [20]. After getting the results from the measurements, it is crucial to map data values that are processed by the device under test (DUT) into predicted power consumption values, so that later on it may be possible to compare them with actual power consumption values. Power consumption models (also called leakage models) allow doing just that. The most known models are the Hamming-Weight (HW), and Hamming-Distance (HD)[16]. The Hamming weight model is the simplest power model. It is based on the assumption that a bit set to 0 does not lead to significant power consumption, while a bit set to 1 does. As such, this model describes the power consumption of a device, at a given moment, as the number of bits in the data being processed that are set to 1.

3.1.1. Attacks based on energy measurement acquired by software

Until recently, power analysis attacks have had two limitations [13]:

1. Instead of aiming at the more complex and high-performance desktop and server CPUs, that kind of analysis aims at small embedded microcontrollers;
2. No software-based attack has been successfully applied, until now, on x86 to catch cryptographic key bits.

There are not many scientific studies regarding vulnerabilities introduced by Intel's RAPL and similar features, as this area is a relatively recent case of study. More recently, O'Flynn *et al.* [17] was able to reach secrets treated in the secure world on an ARM TrustZone-M platform using an onboard ADC, and Mantel *et al.* [14] could differentiate RSA keys by measuring the power consumption on Intel processors. Even more recently, Moritz Lipp *et al.* [13] showed that software-based power SCAs are very powerful against INTEL Software Guard Extensions (SGX), examining the Side-Channel leakage and being able to demonstrate the existent difference between instructions, the Hamming weight of operands and other data, recovering AES-NI and RSA keys in unprivileged and privileged attack scenarios.

3.1.2. Deep Learning Based Profiling SCAs

Analysing the most recent trend in the ML area, the recent works have focused more on DL algorithms such as MLPs [15] or CNNs [6]. Martinasek *et al.* have compared methods based on MLP against more classical SCA approaches such as templates attacks or Stochastic Attacks.

The target is an unprotected implementation of the AES-128 algorithm running on a PIC 8-bit microcontroller.

These studies show that techniques coming from ML are worthy alternatives when compared with the original profiling attacks published in [9] and can often surpass them. It was additionally proved that ML techniques could be used with power traces that have large dimensions, while for the classical approaches it is difficult when the power traces have dimensions greater than 100 and, furthermore, they are robust to signal deformation. The counterpart to the great efficiency of these attacks is that they are difficult to parameterize, which has slowed down their deployment in the security evaluation industry. Moreover, while the papers present promising attack results, they do not give precise information about the parametrization of the algorithms, not even about their training. This is an important limitation which does not allow for the reproducibility of the analyses and hence hampers the development of Deep Learning approaches in the embedded security community. More generally, a common framework to study and compare the effectiveness of Machine Learning methods against embedded implementations of cryptographic algorithms was missing. To surpass this limitation, a common dataset was created.

Based on the new common dataset, some experiments were done by Soroor Ghandali *et al.*[8] ending up with a concrete new architecture called Deep K-TSVM for profiled side-channel attacks which includes two main modules: 1) the Feature extraction module and 2) the classification module.

This uses a Deep Learning architecture of Restricted Boltzmann Machine (RBM) to map datasets and also to learn the pattern of inputs, and it uses TSVM as the classifier. The deep neural network training was separated in two different stages:

1. Pre-training the parameters with RBM using the captured power traces from the copy version of the victim device;

2. Fine-tuning all parameters using gradient descent.

Soroor Ghandali *et al.*[8], claim that Deep K-TSVM performs better compared to the state-of-the-art profiling attacks such as MLP and CNN based side-channel attacks.

Ghandali *et al.* [8] created a new Deep Learning architecture called "Deep K-TSVM" consisting of In and Out and several hidden layers, in which the output of the last hidden layer is taken as the input to the TSVM to find two decision boundaries. It is assumed by the authors that the leakage model is based on the label that is achieved by calculating the HW of the output of the S-Box values at the first round which generates nine different classes, or achieved based on ID classification model (originating 256 classes) of the AES-128. All models predict a class by showing a score value, and the maximum of each prediction's score is computed to unify the class label. In this specific training procedure, 2^n classes were defined for the model, where n is the number of bits to create the ID classification model. The value of the learning rate, number of hidden neurons, and dropout, which have a serious impact on the accuracy and performance of the model, is set before the learning process begins by the grid-search method.

Also, Ryad Benadjila *et al.*[4] performed several tests using MLP and CNN, in order to beat and improve the SCA. Their results show that in the context of SCA CNN are more effective than MLP even if they are more difficult to train.

4. Methodology

This research supported by DL techniques aims at exploiting the correlation between the power consumption and the data processed in a CPU, in particular while performing cryptographic operations. To complete the primary objective of this work two main phases, need to be outlined:

1. Collecting data through software;

2. Evaluating the data with DL models.

For the first main phase, based on previous works [5] and research in Section 3.1.1, it was decided that instead of using a specific framework for power acquisition, the RAPL feature that comes in every Intel processor will be used directly to read the custom kernel module (the MSRdrv), thus the need for an intermediary program, like PAPI [1] or Powercap [2], is eliminated. This choice is expected to minimize the power consumed by the CPU, and decrease the time needed to perform the acquisitions. For the second phase of this work, it was defined that besides the models that have already been used, new models will also be applied that, as shown in Chapter **??**, might get promising results. In this work we will try to successfully apply CNNs, providing them with the hypothetically necessary and suitable layers, and the best performing hyper parameters. It is our objective to also implement a different type of neural networks, RNNs, specifically because it is a type of network that has shown a good ability to recognize patterns in temporally related data.

5. Results

5.1. Discovering the Sampling Rate

Only after discovering the sampling rate and reading rate used for the acquisitions, it is then possible to give the correct parameter to the software used. In this case, we tried using two different machines: 1) a desktop with an Intel I7 10^{th} generation processor with 3.8 GHz clock frequency and 2) an older laptop with an Intel I7 6^{th} generation with 2.6 GHz clock frequency. Looking at Figure 1 it is possible to see many instances where the sampling interval was larger than 100 μs, which results in an average sampling interval of 141.4 μs, even though there seems to be a substantial number of sampling intervals smaller than 100 μs. As such, the average sampling rate acquired in the desktop machine was 7.1 KHz. In this desktop machine, the average reading interval that was obtained during this

experiment was 7.9 μs, which results in an average reading rate of 125.8 KHz. Analysing the Figure 2, it can be seen that the sampling intervals in the laptop machine have a much lower distribution, and most of these intervals are smaller than 100 μs, which was not the case for the desktop machine. In the laptop machine the average sampling interval is 73.1 μs, which results in an average sampling rate of 16.7 KHz, and the average reading interval is 2.7 μs, which results in an average reading rate of 365.0 KHz.

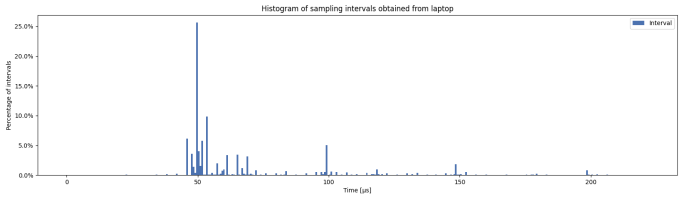

Figure 1: Sampling Interval Histogram acquired on the desktop. Average Sampling Rate of 7.1 KHz and Reading Rate of 125.8 KHz

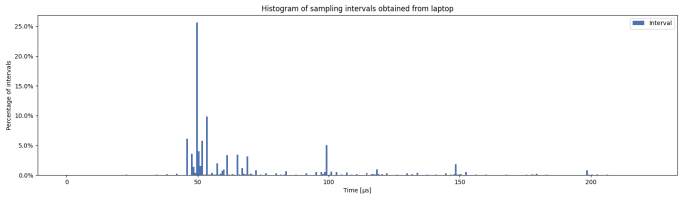

Figure 2: Sampling Interval Histogram acquired on the laptop. Average Sampling Rate of 16.7 KHz and Reading Rate of 365.0 KHz

Given the above data, we opted to continue this work and perform the acquisitions in the laptop machine due to its lower average sampling interval when compared to the desktop. Going forward, the sampling rate chosen to perform the acquisitions in the laptop was 100 μs, which is slightly higher than the calculated average sampling interval, but as can be observed in Figure 2, this sampling rate provides a larger safety margin since the majority of the values in Figure 2 are below this threshold.

5.2. Simplified AES algorithm

The use of a simplified AES provides the base work for trying to evaluate different DL methods and NN hyper parameters, with the objective of gradually increasing the complexity towards a real-case scenario. It is structured in the following way:

1. To increase the leakage of the S-Box all the bytes of the plaintext and all the bytes of the key are the same and previously defined.
2. It performs only the first two operations of the first round of the AES, instead of the regular complete ten rounds, which provides the conditions to explore the leakage of the first S-Box without the following operations and rounds obscuring it.
3. As previously stated, the bytes will be represented with the Hamming weight model, since in this case they can only assume 9 different values.

   The following experiments will gradually increase the possible values of Hamming weight available:

   (a) two levels (0 and 8);

   (b) three levels (0, 4, and 8);

   (c) five levels (0, 2, 4, 6, and 8);

   (d) nine levels (0, 1, 2, 3, 4, 5, 6, 7, and 8).

In order to get the output values of 0x00, 0x01, 0x03, 0x07 0x0F, 0x1F, 0x3F, 0x7F and 0xFF, it is necessary to get the values 0x52, 0x09, 0xD5, 0x38, 0xFB, 0xCB, 0x25, 0x6B and 0x70 from the previous calculation. With the established value 0x50 for the key it is necessary to vary the value for the plaintext such that $p \oplus$ 0x50 = 0x00, $p \oplus$ 0x50 = 0x01, $p \oplus$ 0x50 = 0x03, $p \oplus$ 0x50 = 0x07, $p \oplus$ 0x50 = 0x0F, $p \oplus$ 0x50 = 0x1F, $p \oplus$ 0x50 = 0x3F, $p \oplus$ 0x50 = 0x7F and $p \oplus$ 0x50 = 0xFF, which results in 0x02, 0x59, 0x85, 0x68, 0xAB, 0x9B, 0x75, 0x3B and 0x2D respectively, to produce those values.

5.3. Collecting and Processing Data

For this dataset, we opted for separating the total number of acquired samples into blocks of a certain size, in order to create numerous smaller and distinct power traces for each acquired Hamming weight value, instead of a single large power trace for each HW value. There are a few important and necessary steps to take into account considering the acquired power consumption values:

1. Correcting all the null values that might have been read from the register;
2. Applying a moving average in order to eliminate eventual spikes and other disparities;
3. Separating the data in smaller power traces with 700 or 900 power samples per power trace;
4. Assigning a label to each power trace, the Hamming weight value at the output of the S-Box operation;
5. Making sure that all the labels have the same number of power traces in order to have a balanced dataset;
6. Randomizing the order of the created smaller power traces, in order to have get the most accurate results while training the NN;
7. Dividing the power traces into two groups, with 70% comprising the training and validation set, and 30% comprising the testing set;

5.4. Creating and Choosing the DL Model

The next step is the implementation of the NNs. Several different NNs architectures will be implemented:

1. The CNN, trained with 4, 5, 6, 7, and 8 layers, with 64 kernels per layer;
2. The RNN, trained with 2, 3, and 4 layers, with 100 units each;
3. The ConvLSTM, trained with 1, 2, and 3, with 256 kernels each.

During this training process there are various hyper parameters that can be modified, and each hyper parameter can have a bigger or smaller impact during training. Based on previous smaller quick tests and past research [4, 8], some hyper parameters were defined and fixed for all the experiments. For the model evaluation, the metrics used were: key rank, confusion matrix, and accuracy.

5.5. Two levels of Hamming weight

Following the explanation in Section 5.3, for the two most distant levels of HW, we acquired in total 3502397 samples. In each dataset the data was divided into two major groups of power traces, training and testing. In Figure 3 a clear difference of the power spent when loading the different values from the S-box, with the loading of the bytes with value 0xFF consuming more than the bytes with value 0x00, as expected, since the CPU should spend more energy processing bits of value '1', rather than bits of value '0'.

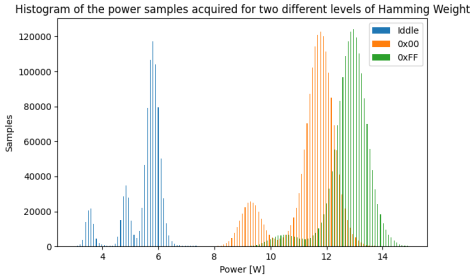

Figure 3: Power Histogram of Loading Idle vs 0x00s vs Loading 0xFFs.

After all the different architectures that were tested, the best model ended up with an overall accuracy of 99.73%, with the 4 layers CNN architecture, being possible to study not only the accuracy looking at the Confusion Matrix, but also the behavior of the trained model through the ROC Curve.

This accuracy can be analyzed more specifically:

- For the level zero of HW the model was able to correctly predict 99.73 % of the time, with only two misclassified cases;
- For the level eight of HW the model was able to correctly predict 99.73 % of the time, also with only two misclassified cases.

These results show that the model was capable of almost perfectly distinguish all the power traces, only missing 4 power traces in total of the testing dataset. The ROC curve reflects the absolute confirmation of the success rate of this model, not only because the curve is standing on the true positive rate side but also because the area is equal to 1, which equates to the best possible performance.

5.6. Nine levels of Hamming weight

For our last experiment, we tried to classify all the nine different classes. Were acquired 11090750 samples and divided in nine major groups of power traces, one for each label.

5.6.1. Training and Results

The confusion matrix of the best performing model, the CNN with 7 layers with an accuracy of 81.10% was calculated. More specifically:

- For the level one of HW the model was able to correctly predict 98.10 % of the times, meaning that in 420 power traces the model correctly classified 412 of them, which was the class with the best accuracy result.
- For the level two and three of HW the model was able to correctly predict 60.77 % and 64.29% of the times. It resulted in the worst obtained results. The main cause for the poor predictions on level two of HW was the improper classification of 82 power traces as level one of HW, and 65 values as level zero of HW. When analysing the predictions of level three it is clear that it went wrong by misclassifying this level as all the HW levels lower than itself, with 164 bad predictions in a total of 170.

Even though the obtained results are not as good as expected, the general division per classes made by the model was quite good. It is possible to see that when the model is classifying power traces that have bigger levels of associated HW, it does not ever predict a value from the smaller levels of HW. Almost the same happens when predicting the smaller levels of the HW, the model only predicts three values corresponding to the bigger levels of HW. So, generally, the model mostly misclassifies power traces with HW values close to the real value.

5.7. Key Rank

The Figure 4 shows the evolution of the Mean Rank using this model, given successive power traces for classification. As it can be observed, the model was able to achieve a Mean Rank equal to zero after only 5 traces, which means that with simply 4500 samples of the testing dataset this model was able to completely recover one byte of the secret key.

This model can be used to discover any secret key byte through the power consumption of a CPU acquired in the same conditions as the training dataset.

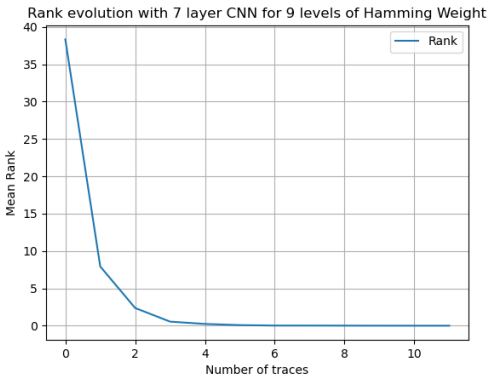

Figure 4: Evolution of the Mean Rank of the attack performed with the best CNN model trained in Section 5.6.

5.8. Results and Analysis

All the gathered information was gathered and displayed into two different tables:

- The CNN results (Table 1), where all the information regarding the accuracy of the trained CNN models is organized;
- The RNNs and ConvLSTM (Table 2), where all the information regarding the accuracy of the RNN and ConvLSTM trained models is organized.

The overall obtained results shown in Table 1 show good accuracies and did not evolve in a completely uniform way, but it is possible to see that as the complexity increased, the models with fewer number of layers started to have poor results. At the same time, when the complexity was lower and a model with more layers was used, the accuracy also suffered.

The CNN with four layers had a great performance when training and evaluating the simpler dataset, achieving the best accuracy result for the two different levels of HW. Also, the seven layers CNN was able to achieve the same result, with both results being reached using 700 samples per power trace. For the three different levels of HW the CNN with best accuracy was the one with six layers and 700 samples per power trace. For the five and nine different levels of HW the model with the best accuracy results was the seven layer CNN, using 700 samples for the five different levels and 900 samples for the nine different levels.

One aspect that is immediately evident is that all the results using five and eight layers were never the best results. In fact, the majority of the calculated accuracies seem to improve from using four layers to five layers, and then all accuracies are worse with eight layers rather than seven layers. This shows that for the more complex datasets, CNNs with fewer layers cannot handle the complexity and the small differences that exist between power traces of different HW values. However, if the CNN complexity is increased too much (as in the eight layer CNN), the created model might be prone to overfitting to the training data, because of all the available training parameters. As such, the more complex models all show poor results in the testing datasets. All the best accuracy results are concentrated between six and seven layers, being the seven layer architecture the one with consistently better results. Overall, the models with fewer layers cannot deal with the more elaborate datasets, and the model with more layers adapts too much to the training set, making it ineffective in the training stage, or the attack stage.

Table 1: Accuracy results for different trained and tested CNN configurations.

Hamming Weight	Samples/ Power Trace	CNN Layers 4	5	6	7	8
2	700	**99.7%**	99.1%	99.5%	**99.7%**	96.4%
	900	87.5%	99.0%	99.2%	98.4%	96.0%
3	700	90.3%	90.8%	**94.3%**	93.2%	70.6%
	900	93.6%	94.1%	93.6%	93.5%	71.0%
5	700	67.8%	85.2%	83.0%	**88.7%**	40.7%
	900	74.2%	74.0%	87.3%	86.5%	44.9%
9	700	8.6%	71.5%	72.3%	72.3%	11.0%
	900	12.6%	59.9%	67.1%	**81.1%**	16.0%

When talking about the number of samples per single power trace, the best results were achieved when using 700 samples, except for the ninth level for HW where 900 samples resulted in the best accuracy possible. For the difference in using 700 samples or 900 samples per power trace, it seems the best results are almost always with the 700 samples dataset, but looking at individual cases there is no clear pattern that allows us to conclude one dataset is much better than the other.

Moving on to the other type of NNs also implemented in this work, are oftentimes very difficult to train, due their architecture not being strictly feed-forward which makes their signal movements extremely complex and makes it hard for the optimization algorithm to update the weights, which results in poor results. That is perfectly demonstrated in Table 2, where it is possible to see that no accuracy value obtained with an RNN was able to outperform any of the CNNs accuracies. On the other hand the results obtained from the ConvLSTMs for the two different levels of HW was similar to the CNN's, but for the other datasets the accuracy results were quite lower than the ones obtained by the CNN models.

With the model obtained for the nine different levels of HW, the performed SCA was remarkably successful, taking five power traces to discover the key byte, which in this case reveals the whole key, since all bytes are the same. This is in line with the desired result for this experiment, since the full byte was recovered.

5.9. Full AES Towards a Real Case Scenario

Taking into consideration the previous results, instead of using the initial three predefined NNs, only the CNNs and

Table 2: Accuracy results for different trained and tested RNN and ConvLSTM configurations.

Hamming Weight	Samples/ Power Trace	RNN (layers) 2	RNN (layers) 4	ConvLSTM (layers) 1	ConvLSTM (layers) 2	ConvLSTM (layers) 3
2	700	95.2%	73.7%	97.6%	97.2%	**98.0%**
	900	71.7%	93.8%	97.8%	**98.0%**	97.5%
3	700	79.2%	79.0%	85.3%	**86.1%**	84.3%
	900	61.1%	53.1%	82.7%	83.9%	84.1%
5	700	25.2%	19.4%	19.8%	53.7%	54.5%
	900	25.1%	19.9%	19.5%	19.5%	**56.0%**
9	700	11.0%	11.0%	11.2%	**51.6%**	11.0%
	900	22.8%	10.7%	10%	10%	48.2%

ConvLSMTs will be used.

5.10. Two levels of Hamming Weight

For just two different levels of HW, as shown in Figure 5, even with the implementation of one entire round of AES, it is possible to perfectly visually distinguish the values 0x00 and 0xFF coming from the S-box.

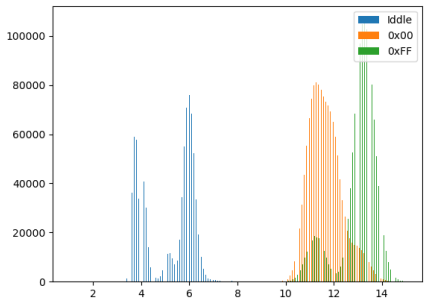

Figure 5: Power Histogram of Loading Idle vs 0x00s vs Loading 0xFFs

5.10.1. Results and Analysis

We selected the accuracy value with the bigger percentage because it is the one with more probability of success, which in this case belongs to the CNN models that uses four layers with 700 samples per power trace. Looking with more detail at the Confusion Matrix, it is possible to notice that:

- For the level zero of HW the model was able to correctly predict 92.53% of the time, only getting wrong the classification of 41 power traces of this level;
- For the level eight of HW was able to correctly predict 84.76% of the time. For this level, the model misclassified 87 examples, mora than for level zero of HW.

The predictions for the eighth level of HW were wrong more than twice the times of the predictions of the level zero of HW. Comparatively to the results achieved in the first part of this work, the NN model accuracy was quite lower, with a difference of 11.2%. The ROC curve reflects the problems caused by the increased complexity to the success rate of the NN model training, not only because the curve is now standing much more towards the center of the true positive rate side but also because of the area that previously was equal to 1 and now decreased to 0.886 meaning a good performance but not the best possible.

5.11. Nine levels of Hamming Weight

The best obtained value came from a ConvLSTM with three layers that achieved 38.7% of accuracy, which was the chosen model to continue the study.

5.11.1. Training and Results

The last experiment shows that:

- For the levels two and seven of HW, the NNs were able to correctly predict 86.50% and 93.74% of the times, respectively, meaning that in 437 and 431 power traces the model correctly classified 378 and 404, respectively.
- For the level five of HW, the NN was able to correctly predict only 6.05% of the time, which was the class with the worst accuracy result. In this case, the main contribution for this low accuracy was that the power traces with level five of HW were more times individually classified as any other level of HW (except level seven), rather than as the level five itself.

Unlike in the equivalent experiment in Section 5.6, this model does not show such a good separation of the power traces associated with bigger HW values and the ones associated with lower HW values. This is expected, as the full first round of the AES algorithm conceals the association between the power trace and the HW value it represents.

5.12. Key Rank

The Figure 6 shows the evolution of the Mean Rank using this model, given successive power traces for classification. As it can be observed, the model was able to achieve a Mean Rank of zero in less than 15 traces, which means that with less than 13500 samples of the testing dataset this model was able to completely recover one byte of the secret key. Moreover with this model the SCA only needs, on average, 7 power traces to recover the key byte.

This model can be used to discover the secret key byte through the power consumption of a CPU acquired in the exact same conditions as the training dataset.

5.13. Results and Analysis

After all the experiments and results presented in this Chapter, the first thing that can be noticed by looking at Tables 3 and 4 is that the global values of the obtained accuracy are lower than the previous ones obtained when applying the simplified AES during the encryption.

It can also be observed, with the exception of the two different levels of HW, which also obtained the best accu-

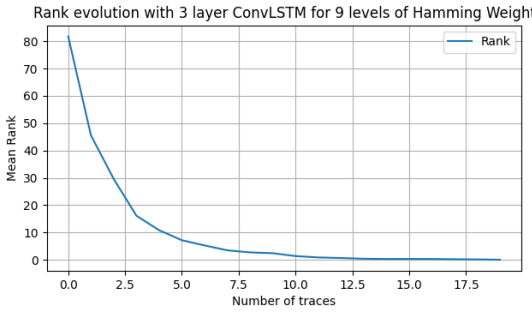

Figure 6: Evolution of the Mean Rank of the attack performed with the best ConvLSTM model trained in Section 5.11.

Table 3: Accuracy results for different trained and tested CNN configurations.

Hamming Weight	Samples/ Power Trace	CNN (layers) 4	5	6	7	8
2	700	**88.6%**	88.5%	88.3%	88.1%	87.7%
	900	87.5%	88.1%	87.5%	86.0%	87.1%
3	700	60.9%	56.7%	29.6%	**66.2%**	65.7%
	900	53.0%	54.0%	60.0%	57.0%	32.5%
5	700	27.2%	18.4%	19.4%	24.2%	24.6%
	900	19.2%	19.0%	26.6%	**30.7%**	30.1%
9	700	25.0%	25.8%	23.1%	**27.7%**	13.2%
	900	19.5%	16.2%	25.6%	18.1%	24.2%

racy value using the CNN with four layers and 700 samples per power trace, that the best results for the CNNs were all achieved when applying seven layers either using 700 or 900 samples per power trace. This shows that the increase in complexity with the full round of AES forces us to generally consider the CNN with seven layer more capable of performing this classification task, and that the simpler architectures do not have enough trainable parameters to capture all the needed information to correctly classify the test power traces. But there is still a decrease in accuracy when training CNN with eight layers, which shows that it is still possible to create models that are too complex for the data they are classifying. What is novel, is the good results obtained by the ConvLSTM networks when compared to the CNNs. For all different levels of HW, except five, the ConvLSTM has performed comparably to the CNN, or even better in some cases. As such, the best accuracy for the full range of HW values with one full round of AES was with the ConvLSTM architecture, using three layers with the 900 samples per power trace dataset. Although this accuracy is much worse than the one obtained with the equivalent dataset with the simplified AES, it is still far from arbitrary classification by the model. It can be concluded that the ConvLSTM architecture does not outperform the CNN for simpler datasets, but that for more complex datasets, the time advantage that this type of network brings is relevant for this problem. For the ConvLSTM, it can also be observed that it mostly benefits from processing a larger time series, that is, processing power traces with 900 instead of 700 samples, which can be explained by its capabilities of dealing with time related data. Finally, the key rank computed in this Chapter also shows very good results, it is only needed less than 15 power traces for a correct key recovery using the best performing model, the ConvLSTM with 3 layers trained with the 900 samples per power trace.

Table 4: Accuracy results for trained and tested ConvLSTM configurations.

Hamming Weight	Samples/ Power Trace	ConvLSTM (layers) 1	2	3
2	700	87.6%	**88.3%**	73.7%
	900	87.3%	87.4%	87.0%
3	700	32.5%	65.1%	68.1%
	900	33.2%	33.6%	**68.4%**
5	700	19.0%	19.0%	19.1%
	900	19.2%	**20.4%**	19.2%
9	700	10.9%	10.3%	11.0%
	900	10.9%	10.3%	**38.7%**

6. Conclusions and Future Work

By analysing the obtained preliminary results coming from the power traces of encryptions under these conditions, which represents the link between the processed data and the power consumption detected with RAPL, we were able to conclude that a leakage exists and is possible to detect, and that could possibly be exploited in a manner that might put at risk the system security.

This work can be divided into two main parts, following the steps that were just mentioned. The first part, where it was proven that a leakage can be detected through software based acquisition, and a key byte can be discovered from a simplified encryption implementation, the simplified AES, which we were able to do with great success.

The second part of this work was focused on increasing the complexity by implementing the Full AES algorithm with only one round, maintaining the key and the plaintexts as they were. All the other processes were exactly the same as before, including the NN models used with the exception for the RNNs that, as shown before, were not suited for this specific task. As expected, the results obtained for this experiment were not as impressive as the first ones, specially for the higher levels of HW, where the accuracy values were much lower. This reduced capability for good predictions, was certainly due to the considerable amount of extra operations introduced by the full round of AES encryptions, that masked correlation between the power consumption and the value of the byte under attack. Despite this, the experiment showed that, under the testing conditions, it is possible to detect a leakage and use it to discover a key byte from a full 16 byte AES key with seven power traces, on average, proving that it is pos-

sible to use software based tools for measuring the CPU power and also use NN models to correlate, evaluate and formalize the attack, albeit in a simplified scenario.

In our work, some processes could have been done differently, such as applying more preprocessing to the data before creating the datasets, namely a z-score normalization which is commonly done in this type of work, and varying the trained NN hyper parameters, specially the loss function, the optimizer, the learning rate, and if the used GPU had more available memory, the batch size could be incremented speeding up all the processes. Although, given the complexity of the work done and the achieved results, those processes turned out not to be necessary to implemented.

There are more steps that can be taken towards the achievement of a complete full AES SCA that more closely relates to a real world scenario, that were not implemented in this work. These steps would be, performing the AES algorithm in its totality, with the ten complete rounds (even though this objective may also be divided in smaller incremental steps), randomizing all the 16 bytes of the plaintext or the key and randomizing all the 16 bytes of both the plaintext and the key.

References

[1] Papi. `https://icl.utk.edu/papi/`. Accessed: 2022-01-05.

[2] Powercap. `https://www.kernel.org/doc/html/latest/power/powercap/powercap.html`. Accessed: 2022-01-05.

[3] rapl. `https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html`. Accessed: 2022-01-05.

[4] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas. Deep learning for side-channel analysis and introduction to ascad database. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020.

[5] A. L. N. da Silva. Hacking the systems from within: Using rapl for power analysis. Master's thesis, Instituto Superior Técnico, January 2020.

[6] W. Fischer and N. Homma. *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529. Springer, 2017.

[7] H. Gamaarachchi and H. Ganegoda. Power analysis based side channel attack. *arXiv preprint arXiv:1801.00932*, 2018.

[8] S. Ghandali, S. Ghandali, and S. Tehranipoor. Deep k-tsvm: A novel profiled power side-channel attack on aes-128. *IEEE Access*, 9:136448–136458, 2021.

[9] B. S. Kaliski, Ç. K. Koç, C. Paar, et al. *Cryptographic hardware and embedded systems–CHES 2002: 4th international workshop, Redwood Shores, CA, USA, August 13-15, 2002: revised papers*. Springer, 2002.

[10] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[11] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

[12] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[13] M. Lipp, A. Kogler, D. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss. Platypus: Software-based power side-channel attacks on x86. In *IEEE Symposium on Security and Privacy (SP)*, 2021.

[14] H. Mantel, J. Schickel, A. Weber, and F. Weber. How secure is green it? the case of software-based energy side channels. In *European Symposium on Research in Computer Security*, pages 218–239. Springer, 2018.

[15] Z. Martinasek, P. Dzurenda, and L. Malina. Profiling power analysis attack based on mlp in dpa contest v4. 2. In *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, pages 223–226. IEEE, 2016.

[16] H. Mestiri, N. Benhadjyoussef, M. Machhout, and R. Tourki. A comparative study of power consumption models for cpa attack. *International Journal of Computer Network and Information Security*, 5(3):25, 2013.

[17] C. O'Flynn and A. Dewar. On-device power analysis across hardware security domains. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 126–153, 2019.

[18] R. Paccagnella, L. Luo, and C. W. Fletcher. Lord of the ring (s): Side channel attacks on the {CPU} on-chip ring interconnect are practical. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[19] T. Popp, S. Mangard, and E. Oswald. Power analysis attacks and countermeasures. *IEEE Design & test of Computers*, 24(6):535–543, 2007.

[20] Y. Zhou and D. Feng. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *IACR Cryptol. ePrint Arch.*, 2005:388, 2005.