LEAKAGE CONVERSION FOR TRAINING MACHINE LEARNING SIDE CHANNEL ATTACK MODELS FASTER

A Thesis

Submitted to the Faculty

of

Purdue University

by

Rohan Kumar Manna

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Dr. Shreyas Sen, Chair

School of Electrical and Computer Engineering

Dr. Kaushik Roy

School of Electrical and Computer Engineering

Dr. Anand Raghunathan

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

Head of the School Graduate Program

ACKNOWLEDGMENTS

I would first like to express my sincere gratitude and thank my thesis advisor Prof. Shreyas Sen for his continuous support towards my research, encouragement, and vast expertise. Moreover, I would also like to thank other professors present in my master's thesis committee: Prof. Kaushik Roy and Prof. Anand Raghunathan, for their intuitive opinions, suggestions, and motivation.

I would like to thank my fellow lab mates in the SPARC lab group: Debayan Das and Josef Danial, for the inspiring discussions, recommendations, and support. Also, I would like to thank my childhood friend Sourjya Roy along with his family, Sutapa Roy and Rwitti Roy for their endless support and making me feel at home. Moreover, I am also thankful to my close friends Jyotsana Jha, Shrihari Sridharan, and Sangamesh Kodge for helping me whenever needed.

Finally, I would like to thank and express my profound appreciation to my father Manishi Nath Manna, mother Ruma Manna, sister Rimita Manna, brother-in-law Anil Surapathi, and grandmother Swapna Chatterjee for their constant support and for helping me in every way possible throughout my years of study. This thesis or achievement would not have been possible without them. Thank you.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

SYMBOLS

ρ_{th}	Correlation Coefficient
δ	Dirac Delta function
μ	Learning Rate

ABBREVIATIONS

IoT	Internet of Things
SCA	Side Channel Attack
ML	Machine Learning
DL	Deep Learning
SNR	Signal to Noise Ratio
ANN	Artificial Neural Network
DSP	Digital Signal Processing
CEMA	Correlation Electromagnetic Analysis
MTD	Minimum Traces to Disclosure
EM	Electromagnetic
CMOS	Complementary Metal–Oxide–Semiconductor
SPA	Simple Power Analysis
DPA	Differential Power Analysis
DEMA	Differential EM Analysis
CPA	Correlation Power Analysis
P-ML-SCA	Power Machine Learning SCA
EM-ML-SCA	EM Machine Learning SCA
HW	Hamming Weight matrix
DUT	Device Under Attack
LMS	Least Mean Square
FIR	Finite Impulse Response
IIR	Infinite Impulse Response

ABSTRACT

Manna, Rohan Kumar Master's, Purdue University, May 2020. Leakage Conversion For Training Machine Learning Side Channel Attack Models Faster. Major Professor: Shreyas Sen.

Recent improvements in the area of Internet of Things (IoT) has led to extensive utilization of embedded devices and sensors. Hence, along with utilization the need for safety and security of these devices also increases proportionately. In the last two decades, the side-channel attack (SCA) has become a massive threat to the interrelated embedded devices. Moreover, extensive research has led to the development of many different forms of SCA for extracting the secret key by utilizing the various leakage information. Lately, machine learning (ML) based models have been more effective in breaking complex encryption systems than the other types of SCA models. However, these ML or DL models require a lot of data for training that cannot be collected while attacking a device in a real-world situation. Thus, in this thesis, we try to solve this issue by proposing the new technique of leakage conversion. In this technique, we try to convert the high signal to noise ratio (SNR) power traces to low SNR averaged electromagnetic traces. In addition to that, we also show how artificial neural networks (ANN) can learn various non-linear dependencies of features in leakage information, which cannot be done by adaptive digital signal processing (DSP) algorithms. Initially, we successfully convert traces in the time interval of 80 to 200 as the cryptographic operations occur in that time frame. Next, we show the successful conversion of traces lying in any time frame as well as having a random key and plain text values. Finally, to validate our leakage conversion technique and the generated traces we successfully implement correlation electromagnetic analysis (CEMA) with an approximate minimum traces to disclosure (MTD) of 480.

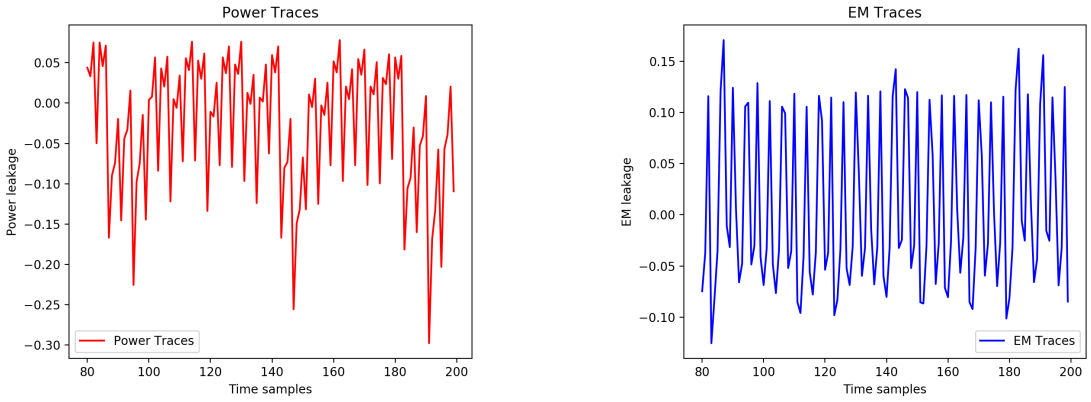
1. INTRODUCTION

Since the invention of computers, their capacity to perform various tasks has gone up exponentially. This consequently has led to the rapid development and improvement of interrelated computing devices. Now, these devices require some protective cryptographic algorithm for communicating safely over the insecure channel or internet. The algorithms stated above provide security to the transmitting data with the help of a secret key value. Hence, a simple stochastic attack on the cryptographic algorithms has a very minute chance of success, which in turn provides an upper hand to the sender and receiver over the attacker. But due to the physical employment of these algorithms information leaks out in the form of power consumed [1], [2], electromagnetic emission's (EM) [3], [4], encryption timing [5], [6], and audible vibrations [7]. The adversary can utilize this freely accessible information to discover the private key from any complementary metal-oxide-semiconductor (CMOS) based embedded device.

1.1 Preliminaries

Almost all CMOS-based embedded devices running encryption techniques are made from simple-logic gates consisting of several transistors. When a charge is applied to the gate of a transistor electrons flow over the silicon substrate. This flow of electrons results in the consumption of power and subsequent production of EM radiations [8]. The power consumed can be measured with the help of Ohm's law [9] and by using a resistor placed in series with the power input. Hence, by dividing the voltage difference across the resistor with the resistor value the power consumed can be measured. Now, these power measurements can be digitally sampled and relocated to a computer for performing power analysis. The EM radiations are collected using

a commercially available EM probe which might be attached to a wide-band low-noise amplifier. The power consumption or EM radiation measured for an encryption process is referred to as a trace. For instance, an encryption operation occurring for one millisecond with a sampling rate of three megahertz will produce a trace with three thousand time points. This thesis will be focusing on only two types of leakage information, power, and EM from an embedded device.



(a) Traces for performing Power Analysis.

(b) Traces for performing EM Analysis.

Fig. 1.1. Waveform of power and EM traces.

1.1.1 Side Channel Attack

Cryptanalytic attacks like the SCA can steal the private key from an encrypted device by using the various types of leakage information [10]. There are two types of SCA attacks, non-profiled SCA and profiled SCA [11], [12]. In a non-profiled SCA attack, the adversary having prior knowledge of the target builds a model by gathering a bunch of the side channel information with known stochastic inputs and anonymously fixed key values. Subsequently, the adversary merges essential assumptions with the help of analytical distinguishers like Person's Correlation to extract the private key value from the side-channel information. Attacks like Simple Power

Analysis (SPA) [13], Differential Power Analysis (DPA) [1], Differential EM Analysis (DEMA) [14], Correlation Power Analysis (CPA) [15], and CEMA [16] fall into the category of non-profiled SCA. However, the success rate of a non-profiled attack is hugely dependent on the quality of the trained model, which is highly vulnerable to the device architecture and software implementation. Moreover, numerous hardware and software countermeasures implementing the technique of varying power consumption have demonstrated high success rates against non-profiled attacks. Thus, profiled SCA attacks dominate over the non-profiled ones.

In a profiled SCA attack the attacker utilizes the side-channel leakage information along with system noise. This attack is performed in two phases. The first phase can be referred to as the training stage where a model is trained with numerous traces having differing sub-key (a small part of the entire encryption key) values. Now in the second phase, the trained model from phase one is used to predict every sub-key of the encryption device being attacked. Thus, by utilizing the system noise for training profiled attacks tend to be stronger against countermeasures like shuffling and masking. Attacks like Power Machine Learning SCA (P-ML-SCA), and EM Machine Learning SCA (EM-ML-SCA) fall into the category of profiled SCA [17].

1.1.2 Power Side Channel Attack

In figure 1.2 [18] we can see how the power traces from an Advanced Encryption Standard (AES) are collected using a power pin. These traces are measured by varying the plain text and by keeping the private key value constant. After the collection stage is complete we move towards the attack stage. For the attack stage initially, a hamming weight matrix (HW) is constructed. This matrix possesses the required power consumption values of the Device Under Attack (DUT) when it performs certain cryptographic operations. Moreover, the matrix is made by keeping the plain text fixed and having all viable combinations of key bytes. Thus, HW helps us to reduce the total possible value for each key byte to 256 ($2^8 = 256$). Now between the power

trace and the power assumption we compute the correlation coefficient (ρ_{th}). Finally, the key byte which separates from the others portrays the maximum correlation and represents the secret key. To reveal all the bits in the AES-128 encryption engine we need to repeat the above-mentioned process sixteen times (as $8 \times 16 = 128$) [19].

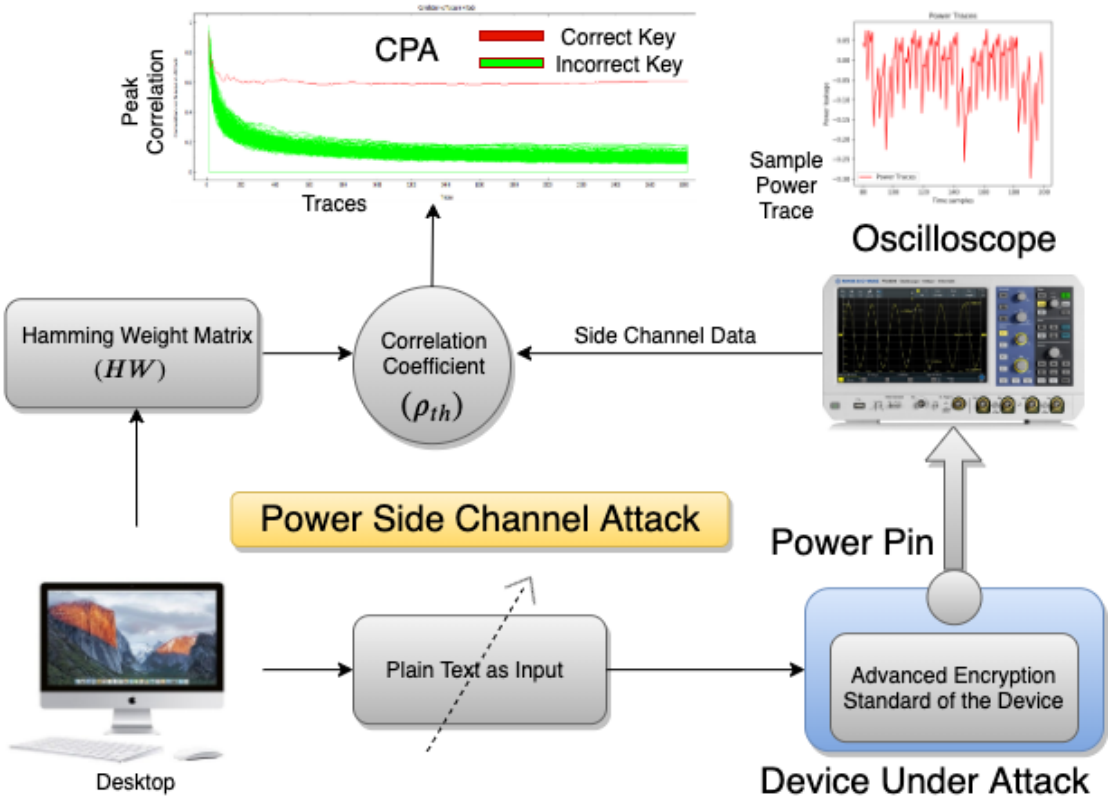


Fig. 1.2. A detailed sketch showing how the secret key can be extracted from power traces.

1.1.3 EM Side Channel Attack

Figure 1.3 [20] provides us with an overview of how the EM traces from an AES-128 encryption engine are collected using a low cost commercially available EM probe. These traces are gathered with the help of an oscilloscope over varying plain text and fixed private key value. Now after the collection stage is complete we move towards

the attack stage. Similar to CPA, for the attack stage initially an HW is constructed which represents a theoretical model. This matrix possesses the required EM radiation values of the DUT when it performs certain cryptographic operations. Moreover, the matrix is constructed by keeping the input plain texts fixed and by having all viable combinations of key bytes. Thus, similar to CPA again HW helps us to reduce the total possible value for each key byte to 256 ($2^8 = 256$). Now between the EM trace and the EM assumption, we compute ρ_{th}. Finally, the key byte which separates from the others portrays the maximum correlation and represents the secret key. To reveal all the bits in the AES-128 encryption engine we need to repeat the above-mentioned process sixteen times (as $8 \times 16 = 128$).

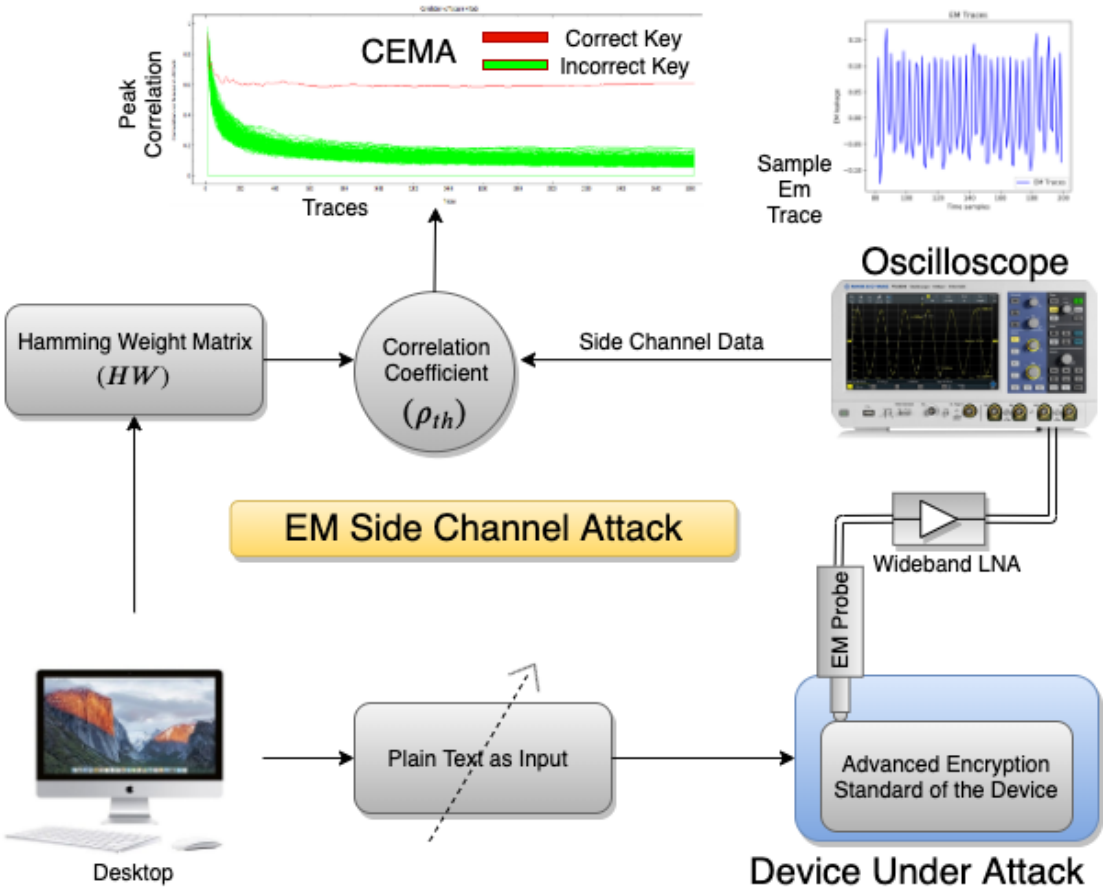


Fig. 1.3. A detailed sketch showing how the secret key can be extracted from EM traces.

1.1.4 Digital Signal Processing Filters

In the field of signal processing, the Dirac Delta function (δ) is referred to as the impulse function. Thus, an impulse function can be defined on the real line having an infinite value at the origin and zero values everywhere else. Now when any dynamic system receives an impulse function it responds to the external changes in a specific way, this response is referred to as an impulse response. Moreover, these concepts are used to make a filter which is a class in the area of signal processing. Thus, a filter [21] can be defined as a mechanism or mathematical process which discards certain undesired components from a signal. There are numerous ways of classifying filters but there is no specific hierarchical order. Hence, for this thesis, we will be focusing only on adaptive digital filters but Table 1.1 given below very briefly describes some of the major types of filters used and their basis of classification [22].

Table 1.1.
Some general types of filters in Signal Processing

Basis for Classification	**Type 1**	**Type 2**
Time Signal	Digital	Analog
Impulse Response	Finite Impulse Response	Infinite Impulse Response
Framework	Discrete-time	Continuous-time
Output Signal	Linear	Non-linear
Dependence on $f(t)$	Time-variant	Time-invariant
Component Used	Active	Passive
Signal Domain	Casual	Non-casual

Digital filters usually increase or decrease certain features of discrete-time signals as opposed to analog filters that operate on continuous-time signals. Mainly there are two types of digital filters, finite impulse response filters (FIR) [23], and infinite impulse response filters (IIR) [24]. A finite impulse response filter can be defined

as a filter having limited impulse response while an infinite impulse response filter typically consists of a feedback mechanism with a decaying response. Now for the first implementation of this thesis, we need to define the Adaptive Least Mean Squared (LMS) filters. Adaptive filters [25] are linear filters having a transfer function with varying parameters. These parameters are governed by the optimization algorithm. Moreover, they utilize error signals in a feedback mechanism to polish the transfer function and reduce the cost for the next cycle.

1.1.5 Artificial Neural Networks

An ANN [26] is a computational mechanism that tries to replicate the way biological neurons [27] function in the human brain. They intend to learn and comprehend data in the same manner as the human brain. In the last decade, artificial neural networks have shown promising results in the area of image processing and pattern recognition due to the vast availability of data and computational resources. However, in this thesis, we will be using ANN for generating averaged EM traces from raw power traces. Mostly, an ANN comprises three important stages which can be described as follows [28].

- Pre-process stage: In this stage, the input data is modified so that it contains only certain key features required during the training stage of the model.
- Training stage: The most important stage in which the weights associated with each neuron is adjusted for reducing the difference between the actual and the predicted result. This adjustment is performed in multiple epochs so that the difference is minimum.
- Validation stage: In this stage, the trained model is used to generate new values. These newly generated values are now compared to the actual values by which the efficiency of the network is measured.

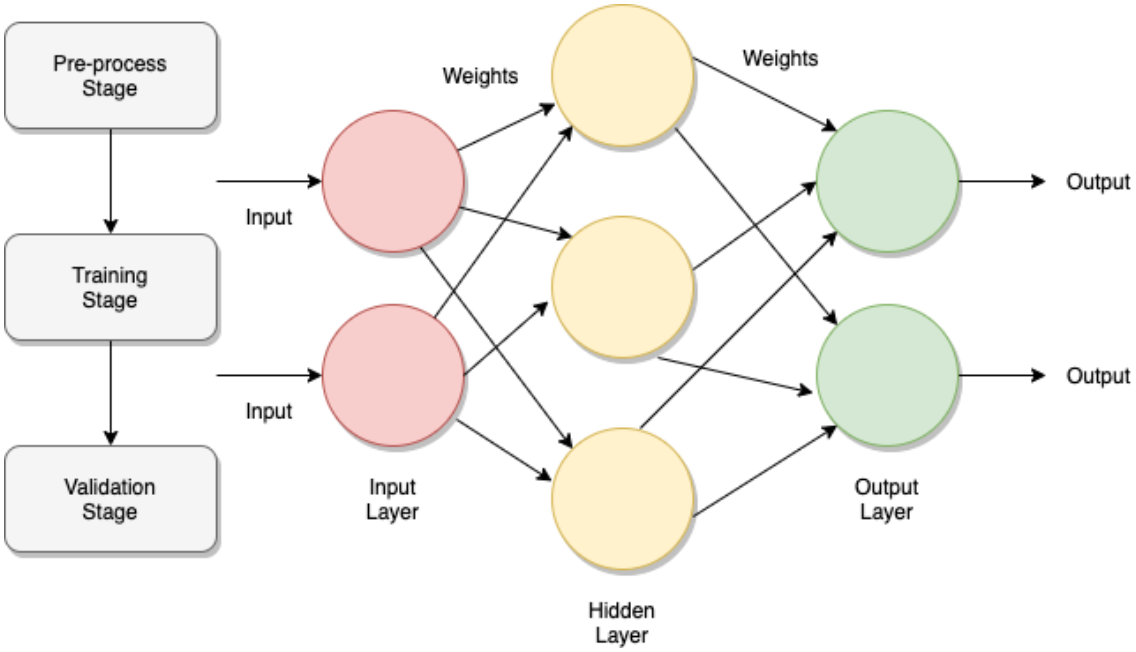


Fig. 1.4. Diagrammatic description of the different stages and the structure of a basic ANN.

Figure 1.4 given above describes the input layer, hidden layer, and the final output layer. The input layer accepts any type of data than initially multiplies the data with random weights and applies the activation function to it for producing output for that layer. Now the output for the input layer is the input to various hidden layers. The Hidden layers further try to learn the non-linear dependencies by following the concept of multiplying inputs with the weights then adding bias and the activation function to it. Finally, the outputs of the hidden layers are the inputs to the output layer. Thus, to summarize mathematically ANN performs the summation of products of the input data and the weights associated with the input, then it applies an activation function to it for producing the output. A few important points to note are that the hidden layers are optional but usually, the network would not learn well without a hidden layer. Moreover, the activation function must be differentiable or else back-propagation cannot be applied for computing gradients. Thus, if gradients cannot be computed for the weights then the weights cannot be optimized by using

an optimization algorithm. Figure 1.5 given below compares an artificial neuron in ANN to an actual neuron in the brain [29].

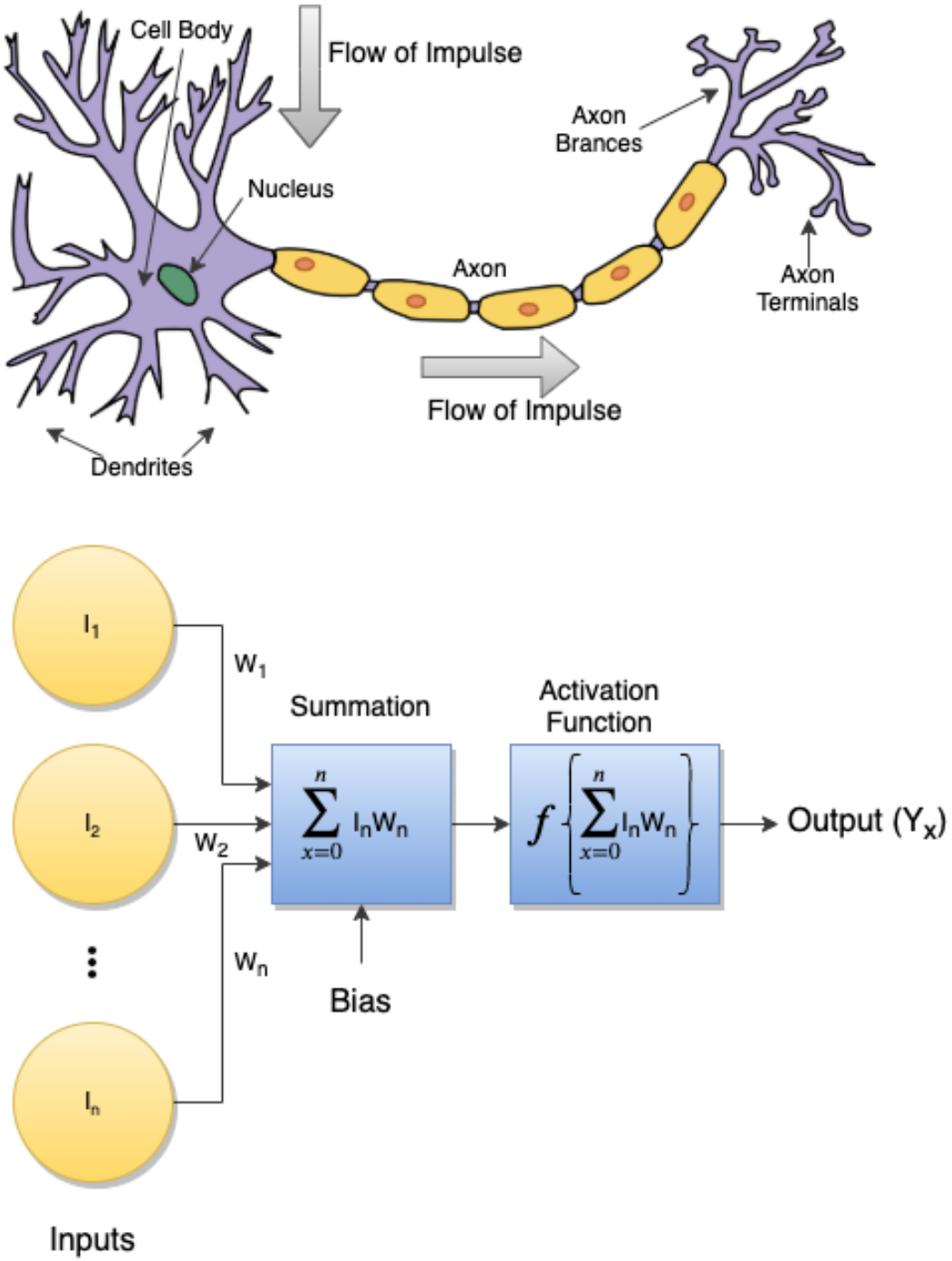


Fig. 1.5. Comparison of an artificial neuron in ANN to a neuron in the brain.

1.2 Motivation

Vast developments in the field of technology have led to an increase in demand for mathematically secure and robust encryption engines. These encryption engines are widely implemented in numerous CMOS-based embedded devices. AES [30] is one of the most popular encryption engines used for encrypting data in most IoT edge devices. However, different forms of SCA attacks like power and EM have become increasingly popular to break these encryption engines and steal the secret information being communicated. In the last decade, machine-learning-based power and EM SCA attacks have shown very promising results and advantages over template attack [31]. Moreover, these machine learning or deep learning-based attacks do not require much statistical analysis for discovering the leakage points. However, as EM traces have low SNR [32] the number of traces required for training the EM-ML-SCA model is a lot and usually, it is difficult to collect that many traces in a real-life scenario. Thus, to solve this issue we try to collect power traces (having high SNR) [32] and convert them directly into averaged EM traces.

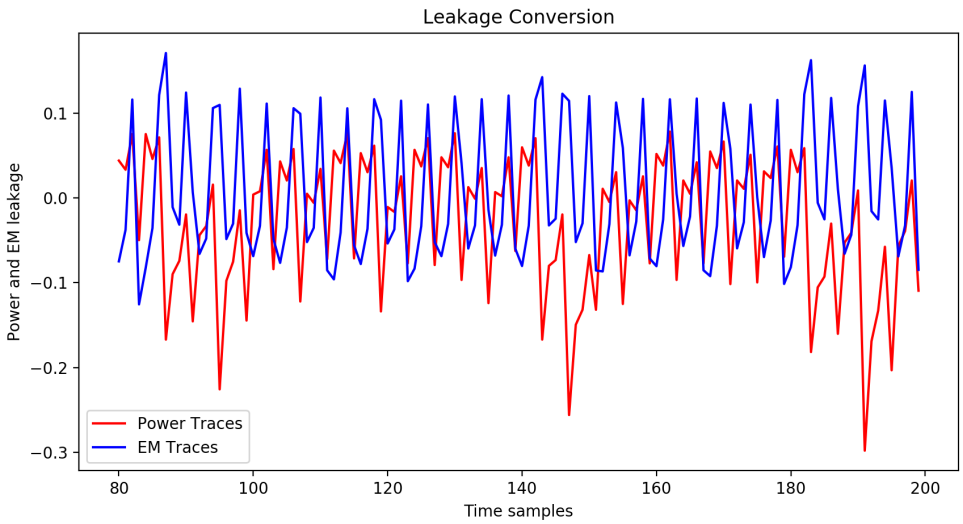


Fig. 1.6. A graph showing how leakage conversion will be conducted.

Figure 1.6 shown above graphically portrays how we plan to convert a value in the power trace to a value in the EM trace. We overlap both the traces on a single graph to show how a particular power value after passing through a general function or model produces the required EM value. This thesis mainly deals with an AES 128-bit encryption engine, different types of DSP filters, different ANN models for conducting leakage conversion, CEMA, and EM-ML-SCA.

1.3 Contribution

In this thesis, we introduce a new technique referred to as leakage conversion where we generate EM traces from power and power traces from EM by using the knowledge and benefits of artificial neural networks. The primary objective of this new technique is to allow faster training of the EM-ML-SCA model by collecting 10X lesser EM traces from the DUT. Hence, the specific and essential contributions of this thesis are briefly outlined as follows.

- At first, we implement the technique of leakage conversion using different types of DSP algorithms. For our implementation, we try to convert power traces to EM. After implementation, we validate the generated EM traces by performing CEMA. Unfortunately, none of the results pass the test but we deduce that adaptive LMS FIR filter of order one produces the best result out of the lot. Now a comparative analysis was performed between the different DSP algorithms and ANN. Results from the analysis show that the ANN model produces much better EM traces than the DSP filters.

- All machine learning models require a lot of data to train. Similarly, the EM-ML-SCA model requires the collection of 10X (X refers to any natural number) traces from the physical encryption device for training with X number of traces. Hence, after showing how an ANN model can outperform any DSP filter, we use ANN to try and convert raw power traces to N-averaged EM traces (where N equals 10). Thus, by collecting one type of leakage information and then

transforming it into an averaged EM trace we show how the EM-ML-SCA model can be trained faster. Consequently, this faster training will lead to faster execution of the profiled SCA attack.

- We validate the authenticity of the generated EM traces by performing the non-profiled CEMA attack with the generated EM traces. Results show the successful extraction of the entire encryption key. Moreover, we also show the successful generation of power traces from EM. However, as EM traces have low SNR, the power traces generated from them do not have enough information for performing CPA.

- Finally, after testing the generated EM traces on the EM-ML-SCA model we encounter a failed profiled attack. Now we decipher the cause of the failure and suggest a new key specific ANN model for leakage conversion. This new technique portrays success to a certain extent. Hence, proving the successful implementation of leakage conversion. Moreover, we also state certain recommendations which might give more concrete results for performing profiled attacks faster.

1.4 Thesis Orientation

The rest of the thesis is oriented in the following order. In section 2, a comparative study between DSP algorithms and ANN is conducted to determine which algorithm gives better generated EM traces. In section 3 we propose the technique of leakage conversion by converting power traces to EM traces and vice-versa. We show how the generated EM traces can be used for non-profiled CEMA and why the generated power traces fail CPA. Finally, we also state how EM-ML-SCA can be trained faster using the technique of leakage conversion. Section 4 very briefly summarizes our discoveries, work done for this thesis, and finally concludes on our mission statement. Section 5 provides some recommendations and highlights the possibility of future research work.

2. DIGITAL SIGNAL PROCESSING FILTERS VERSES ARTIFICIAL NEURAL NETWORKS

2.1 Adaptive LMS FIR Filters

An LMS FIR filter [33] is a type of adaptive filter that tries to find filter coefficients that produce the minimum mean square error between the required signal and the output signal (error signal). The filter follows an iterative optimization method of the cost function (stochastic gradient descent). The adaptive LMS FIR algorithm [34] mainly involves three stages. The first stage involves generating the filter output by using the filter coefficients and the input signal. The second stage involves generating the error by subtracting the generated signal from the desired signal. Finally, the third stage involves adjusting the weights or filter coefficients. This three-stage algorithm is repeated n times for producing the final filter coefficients.

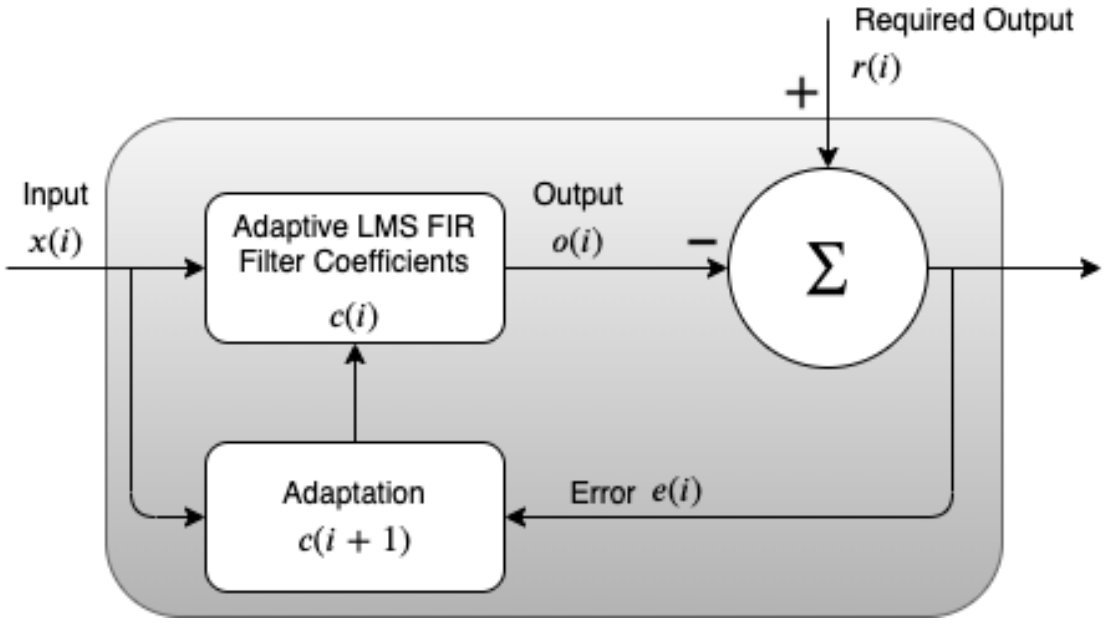


Fig. 2.1. Diagrammatic representation of an adaptive LMS filter.

Figure 2.1 given above shows the control flow of the filter and its various stages but mathematically the three stages can be described as follows:

- Output $o(i) = \sum_{i=0}^{N-1} x(i)c(i)$, where N represents the length of the filter, $x(i)$ represents the input signal, and $c(i)$ represents the filter coefficients or weights.
- Error $e(i) = r(i) - o(i)$, where $r(i)$ represents the required or desired signal.
- Adaptation $c(i+1) = c(i) + \mu \times [x(i)e(i)]$, where μ represents the learning rate.

The following graphs (figure 2.2 to figure 2.7) will now portray our results obtained by performing our proposed technique of leakage conversion (Power traces to Averaged EM traces) using adaptive LMS FIR filters of different orders.

2.1.1 Adaptive LMS FIR Filter of order 0

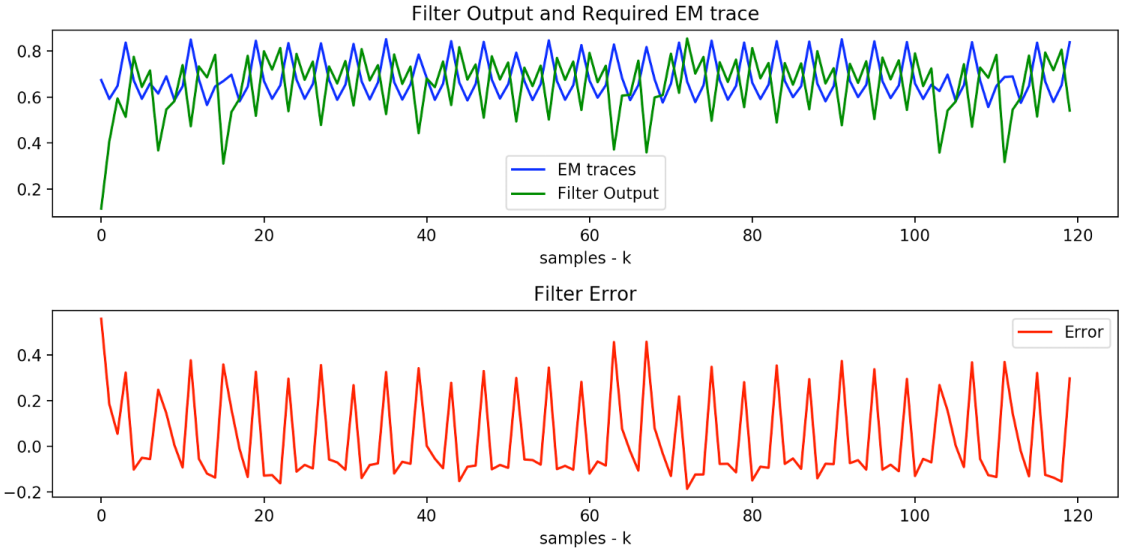


Fig. 2.2. Zeroth order graphical waveform representation of the filter output.

Final Filter Coefficients - [0.83922959] ; $\mu = 0.9$ for key 0 which is averaged over 10000 traces. Concept used : $y(1) = c_0 x(1)$; where $y(n)$ represents the n^{th} EM trace value, $x(n)$ represents the n^{th} Power trace value, and c_n represents the n^{th} coefficient.

From the error curve, we observe that the error ranges between -0.2 to 0.5. Moreover, we also observe that the error range for order 0 is the least when compared to the error range values of the other ordered filters. Thus, we can claim that an adaptive LMS FIR filter of order 0 produces the best possible value.

2.1.2 Adaptive LMS FIR Filter of order 1

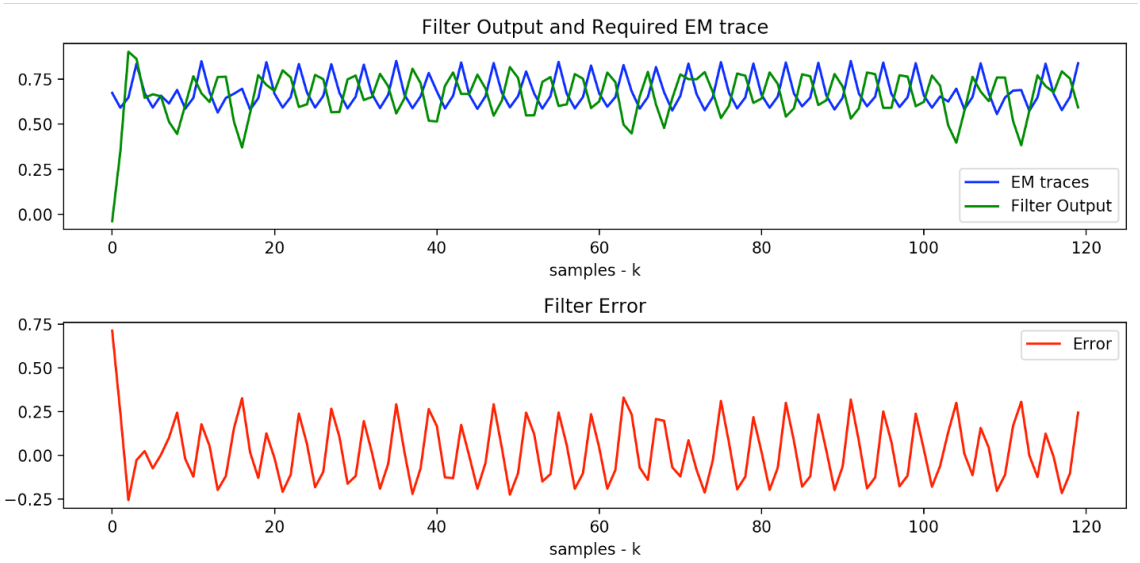


Fig. 2.3. First order graphical waveform representation of the filter output.

Final Filter Coefficients - [0.43308635] [0.35484849] ; $\mu = 0.9$ for key 0 which is averaged over 10000 traces. Concept used : $y(2) = c_0 x(2) + c_{-1} x(1)$. From the error curve, we observe that the error ranges between -0.2 to 0.75. Thus, we can claim that an adaptive LMS FIR filter of order 1 generates more error than a filter of order 0. Moreover, we also observe flattening of the peaks which depict a loss of information during the conversion.

2.1.3 Adaptive LMS FIR Filter of order 2

Final Filter Coefficients - [0.37321845] [0.26339759] [0.17164171] ; $\mu = 0.7$ for key 0 which is averaged over 10000 traces. Concept used : $y(3) = c_0x(3)+c_{-1}x(2)+c_{-2}x(1)$. From the error curve, we observe that the error ranges between -1.0 to 0.6. Thus, we can claim that an adaptive LMS FIR filter of order 2 generates more error than a filter of order 1. Moreover, we also observe more flattening of the peaks which depict a greater loss of information during the conversion.

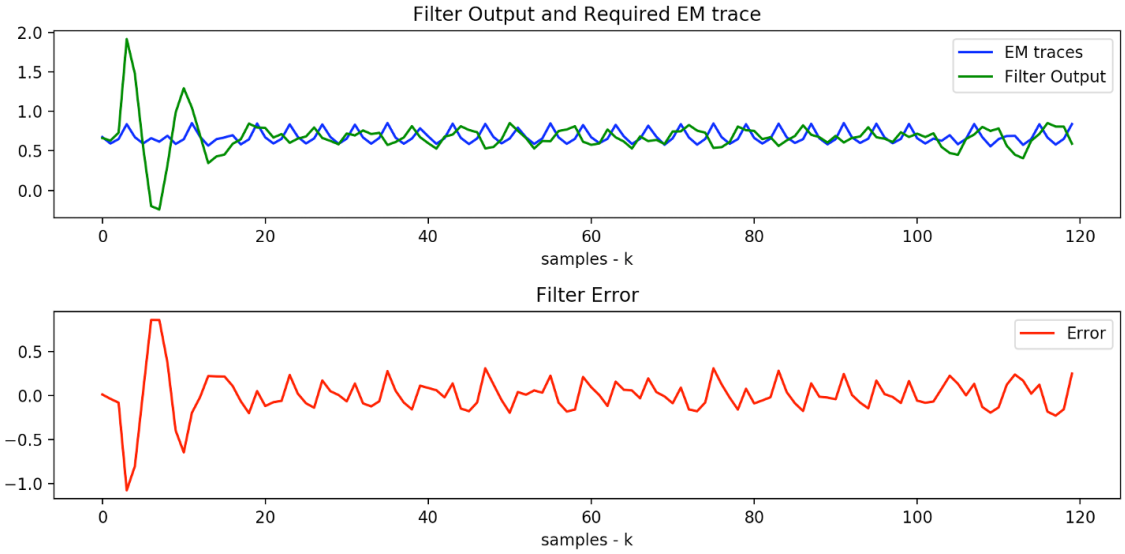


Fig. 2.4. Second order graphical waveform representation of the filter output.

2.1.4 Adaptive LMS FIR Filter of order 3

Final Filter Coefficients - [0.34180162] [0.33443942] [0.27419134] [0.20584869] ; $\mu = 0.4$ for key 0 which is averaged over 10000 traces. Concept used : $y(4) = c_0x(4) + c_{-1}x(3) + c_{-2}x(2) + c_{-3}x(1)$. From the error curve, we observe that the error ranges between -2.0 to 1.0. Thus, we can claim that an adaptive LMS FIR filter of order 3 generates more error than a filter of order 2. Moreover, we now observe

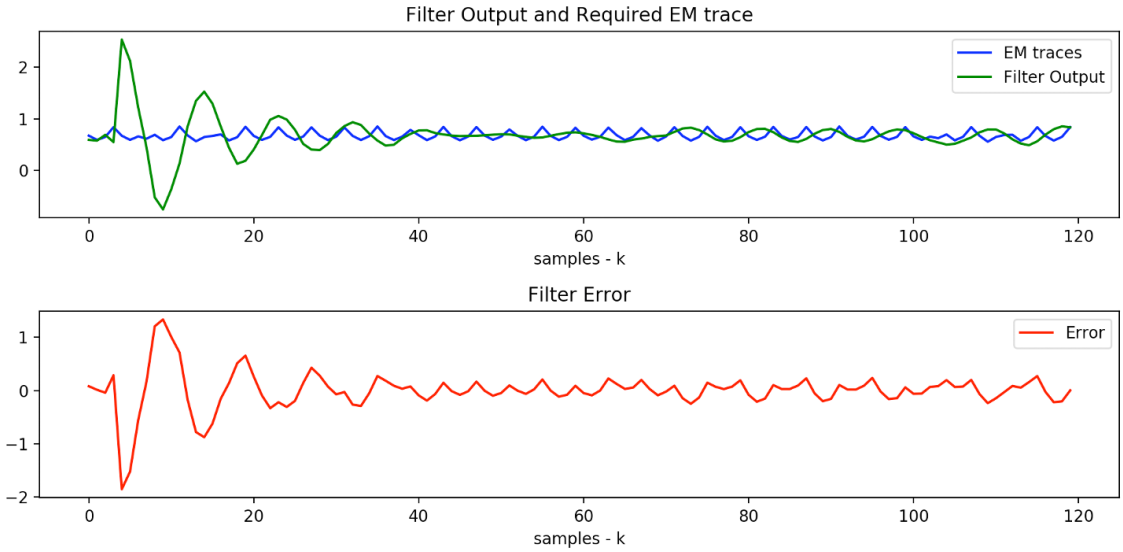


Fig. 2.5. Third order graphical waveform representation of the filter output.

complete flattening of the peaks which depicts a complete loss of information during the conversion.

2.1.5 Adaptive LMS FIR Filter of order 4

Final Filter Coefficients - [0.19140527] [0.23340909] [0.23580694] [0.21169692] [0.18601405] ; $\mu = 0.3$ for key 0 which is averaged over 10000 traces. Concept used : $y(5) = c_0x(5) + c_{-1}x(4) + c_{-2}x(3) + c_{-3}x(2) + c_{-4}x(1)$. From the error curve, we observe that the error ranges between -2.0 to 1.5. Thus, we can claim that an adaptive LMS FIR filter of order 4 generates more error than a filter of order 3. Moreover, we now observe a complete erroneous filter output which has no relation to the desired EM trace in blue. However, we still observe the successful execution of the filter as the output curve converges to some minima even though the results do not meet our expectations.

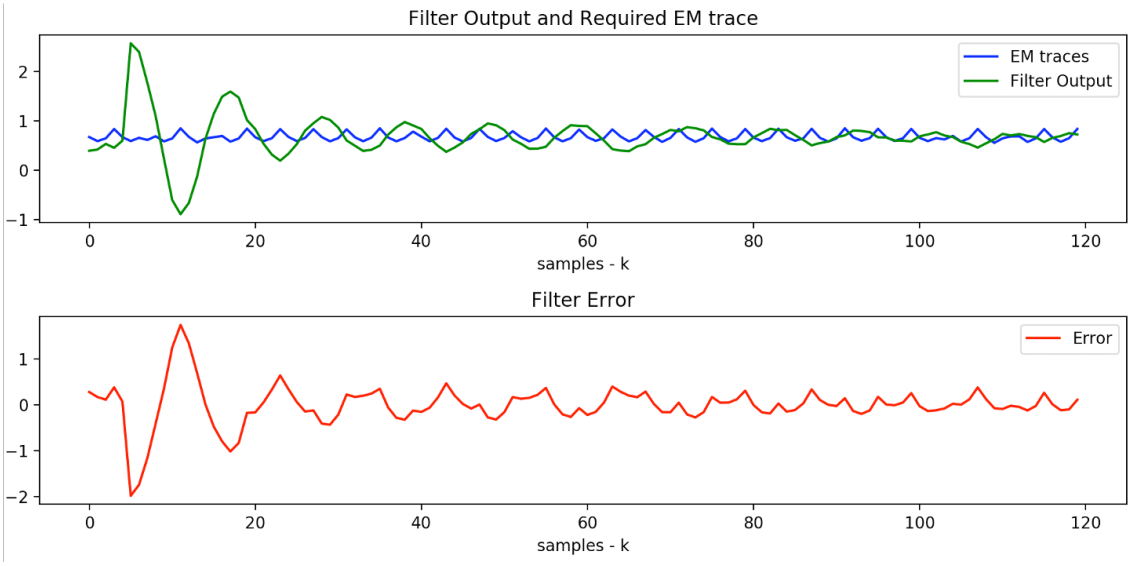


Fig. 2.6. Fourth order graphical waveform representation of the filter output.

2.1.6 Adaptive LMS FIR Filter of order 5

Final Filter Coefficients - [0.17838873] [0.17897847] [0.19031284] [0.19362094] [0.18585556] [0.17730956] ; $\mu = 0.1$ for key 0 which is averaged over 10000 traces. Concept used : $y(6) = c_0x(6)+c_{-1}x(5)+c_{-2}x(4)+c_{-3}x(3)+c_{-4}x(2)+c_{-5}x(1)$. From the error curve, we observe that the error again ranges between -2.0 to 1.5. Thus, we can claim that an adaptive LMS FIR filter of order 5 generates an equal amount of error as a filter of order 4. But, we now observe a almost flat filter output in green which is worse than the output observed for order 4.

Now, there is no point in running filters of order greater than 5 as it would generate more erroneous results. Hence, we can claim that generally digital filters are not a suitable tool for performing the task of leakage conversion as they are unable to learn the non-linear dependencies. This thought leads us to the direction of neural networks which recently have shown tremendous potential in a wide array of problems.

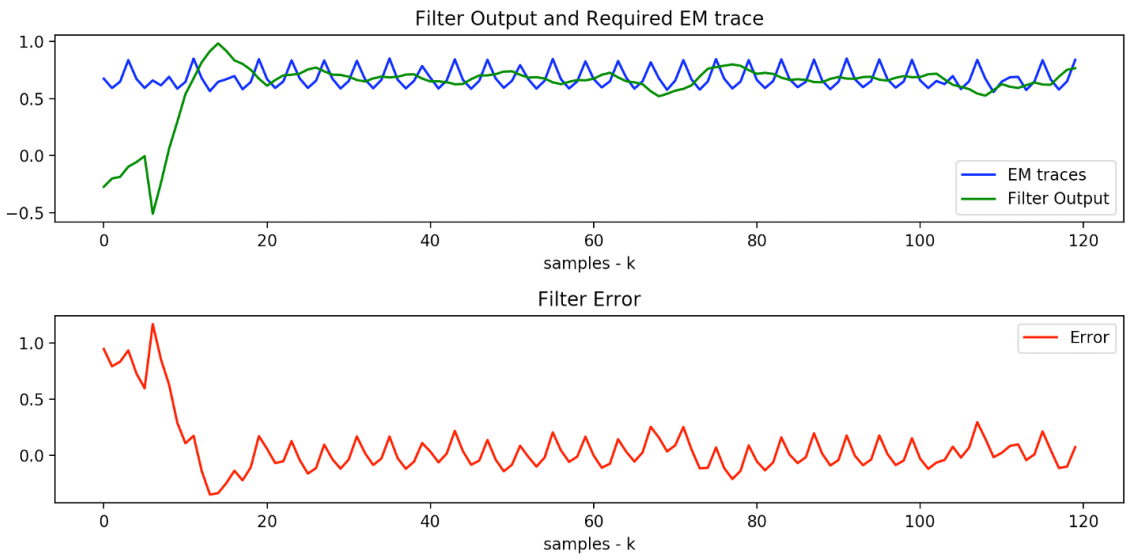


Fig. 2.7. Fifth order graphical waveform representation of the filter output.

2.2 Artificial Neural Networks

ANNs have the unique ability to learn and model complex dependencies between the inputs and the outputs [35]. Moreover, they also possess the ability to generalize [36] without applying constraints on the input data. Hence, we choose ANNs for performing our proposed leakage conversion technique. In this experiment, we aim to find a correlation between power and averaged EM traces taking time samples 80 to 200 into consideration as the encryption operation occurs in that interval. Moreover, we Normalize [37] the data set between zero and one initially just for the sake of testing. The data sets used are as follows.

- 2357_traces.npy – Old 10,000 power traces with 3000 time samples, utilized for training and testing. Moreover, 70% of the data is reserved for training and the rest is used for testing.

- 1516_traces.npy – Old 10,000 EM traces with 3000 time samples, utilized for training and testing. Moreover, 70% of the data is reserved for training and the rest is used for testing.

- 2019.05.11-16.46.40_traces_transfer_power1.npy – Newer 100 power traces with 3000 time samples, utilized as the input while training. Moreover, the traces have the same key and plain text values.

- 2019.05.11-16.40.12_traces_transfer_EM1.npy – Newer 100 EM traces with 3000 time samples, utilized as the desired output while training. Moreover, the traces have the same key and plain text values as mentioned in 2019.05.11-16.46.40_traces_transfer_power1.npy.

- 2019.05.11-16.44.49_traces_transfer_power2.npy – Newer 100 Power traces with 3000 time samples, utilized as the input while validating. Moreover, the traces have the same key and plain text values.

- 2019.05.11-16.42.22_traces_transfer_EM2.npy – Newer 100 EM traces with 3000 time samples, utilized as the desired output while validating. Moreover, the traces have the same key and plain text values as mentioned in 2019.05.11-16.44.49_traces_transfer_power2.npy.

Figure 2.8 to figure 2.16 given below represents the output waveform from the trained neural network model (y-output) in green, the required EM trace waveform (d-target) in blue, and the input power trace waveform (power) in red. The network architecture used is as follows, the input layer consists of 120 neurons for the 120 time samples, the first hidden layer consists of 60 neurons, the second hidden layer consists of 30 neurons, and the output layer again has 120 neurons to generate the averaged EM trace having 120 time samples. Moreover, the model is sequential with all linear layers and no dropout values. The hidden layers have sigmoid activation [38]. For training 80 power traces with 120 time samples and 80 EM traces (averaged by 10) with 120 time samples were selected. For testing 20 power traces with 120 time samples and 20 EM traces (averaged by 10) with 120 time samples were selected. Other hyper-parameters include a learning rate [39] of 0.001, 20 epochs, Mean Squared Error (MSE) [40] as the cost function [41], and stochastic gradient descent [42] as the optimizer. Finally,

if we want to claim that ANNs are a suitable tool for performing the conversion then we need to look at all the possible cases of input data variation. Hence, we now look at the following graphical representations of our results from the various experiments performed with ANNs.

2.2.1 Experiment 1

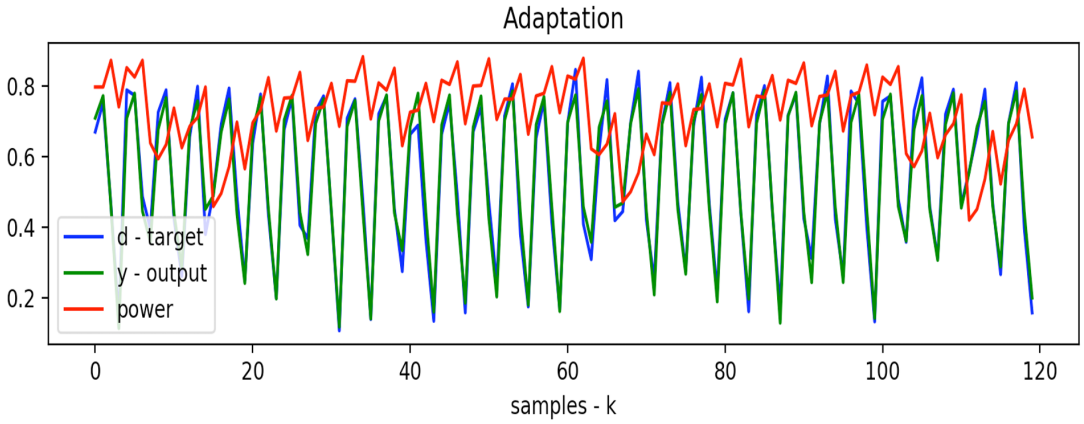


Fig. 2.8. Training and validating on a specific key value with fixed plain text

Here we train and validate with "2019.05.11-16.46.40_traces_transfer_power1.npy" and "2019.05.11-16.40.12_traces_transfer_EM1.npy". Moreover, graphically we can observe a much better result than the resultant curve with filters and the MSE value while testing was observed to be 0.0024.

2.2.2 Experiment 2

Here we train with "2019.05.11-16.46.40_traces_transfer_power1.npy" and "2019.05.11-16.40.12_traces_transfer_EM1.npy" and validate using "2019.05.11-16.44.49_traces_transfer_power2.npy" and "2019.05.11-16.42.22_traces_transfer_EM2.npy". Moreover, the MSE value while testing was observed to be 0.0007 which shows us that the model learns more with more data.

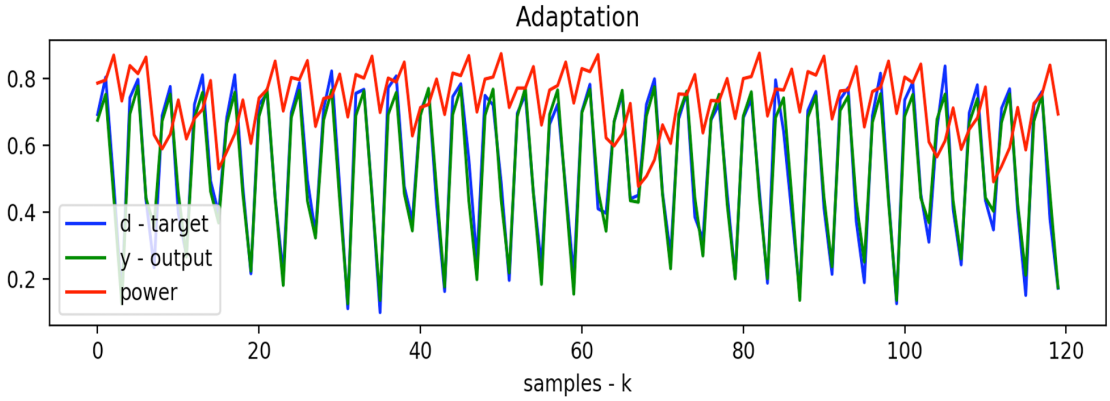


Fig. 2.9. Training and validating on another specific key value with fixed plain text

2.2.3 Experiment 3

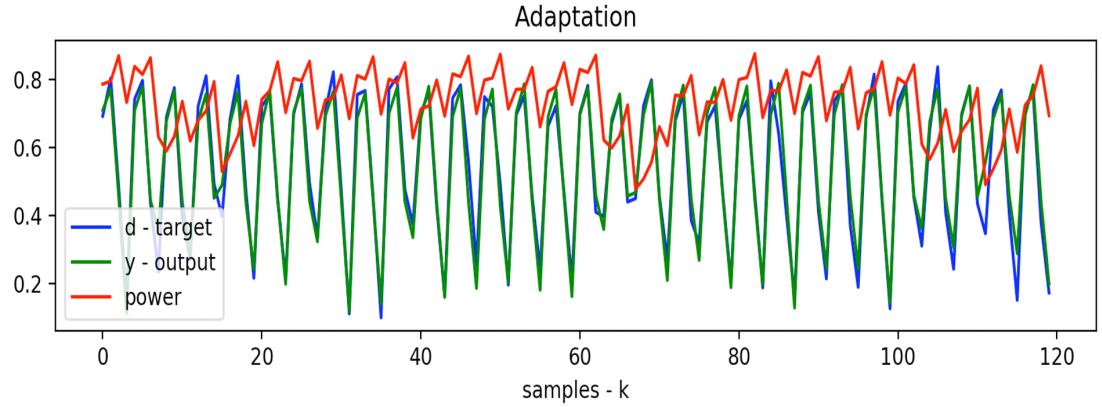


Fig. 2.10. Training on one specific key value but validating on another specific key value while fixing the plain text in both cases

Here we train and validate with "2019.05.11-16.44.49_traces_transfer_power2.npy" and "2019.05.11-16.42.22_traces_transfer_EM2.npy". Moreover, the MSE value while testing was observed to be 0.0090 which is quite similar to the MSE value observed in experiment 1.

2.2.4 Experiment 4

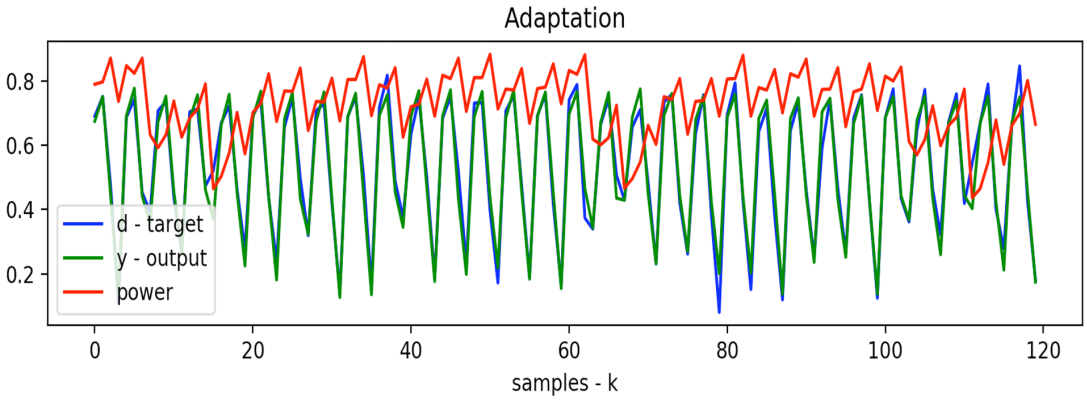


Fig. 2.11. Training and validating on any specific random key value with fixed plain text (Attempt 1)

Here we train with "2019.05.11-16.44.49traces_transfer_power2.npy" and "2019.05.11-16.42.22_traces_transfer_EM2.npy" and validate using "2019.05.11-16.46.40_traces_transfer_power1.npy" and "2019.05.11-16.40.12_traces_transfer_EM1.npy". Moreover, the MSE value while testing was observed to be 0.0020 which is again quite similar to the MSE value observed in experiment 1.

2.2.5 Experiment 5

The training and validation data are the same as mentioned in experiment 4. Moreover, the MSE value while testing was observed to be 0.0021. Hence, we can now test for any random configuration of the input data. However, we also observe that the peaks (green and blue) do not co-inside in many cases. It should be noted that to eliminate the possibility of randomly testing the same key values we perform two experiments (4 and 5) with the same configuration.

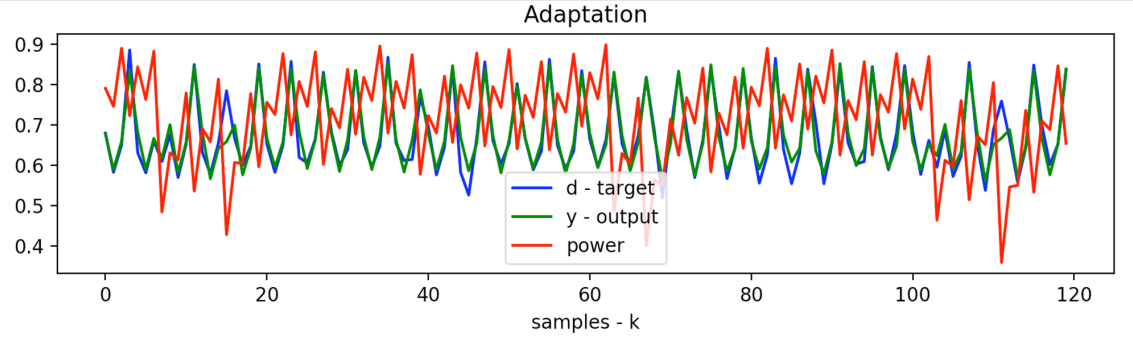


Fig. 2.12. Training and validating on any specific random key value with fixed plain text (Attempt 2)

2.2.6 Experiment 6

The training and validation data are the same as mentioned in experiment 4 but the plain text values are not fixed anymore. Moreover, the MSE value while testing was observed to be 0.0024. Hence, we can now claim that for all the different types of configurations we will surely get a small error value.

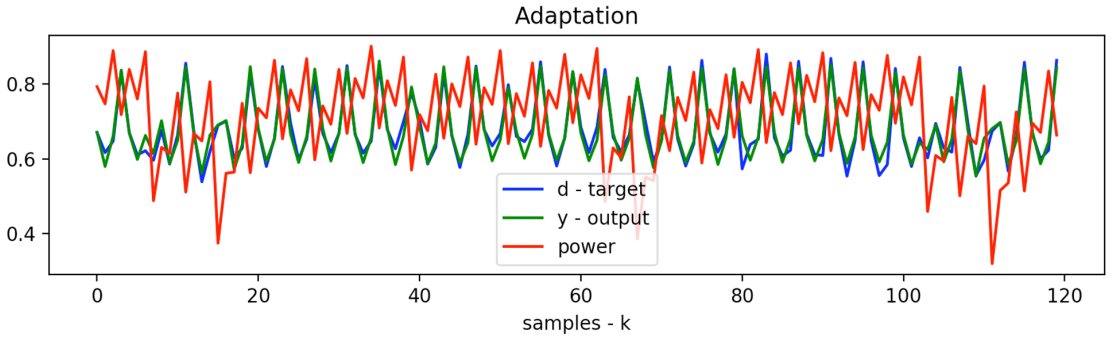


Fig. 2.13. Training on any random specific key value but validating on another random specific key value (Attempt 1)

2.2.7 Experiment 7

The training and validation data are the same as mentioned in experiment 4. Moreover, the MSE value while testing was observed to be 0.0023. Again, it should be noted that to eliminate the possibility of randomly testing the same key values we perform two experiments (6 and 7) with the same configuration.

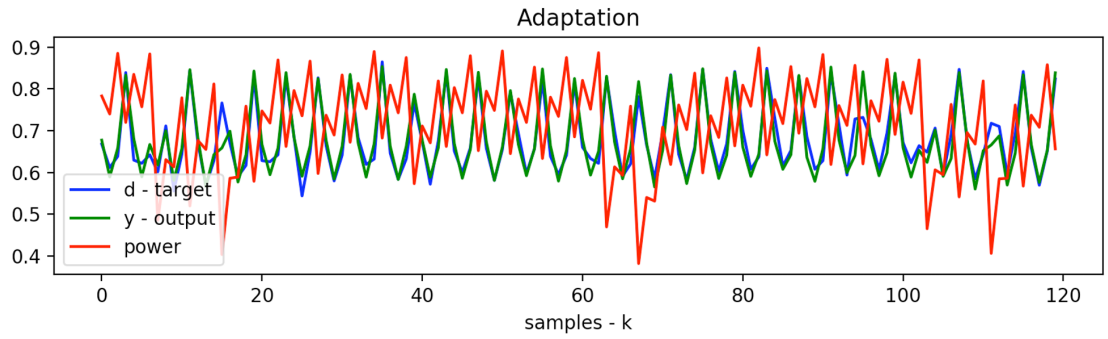


Fig. 2.14. Training on any random specific key value but validating on another random specific key value (Attempt 2)

2.2.8 Experiment 8

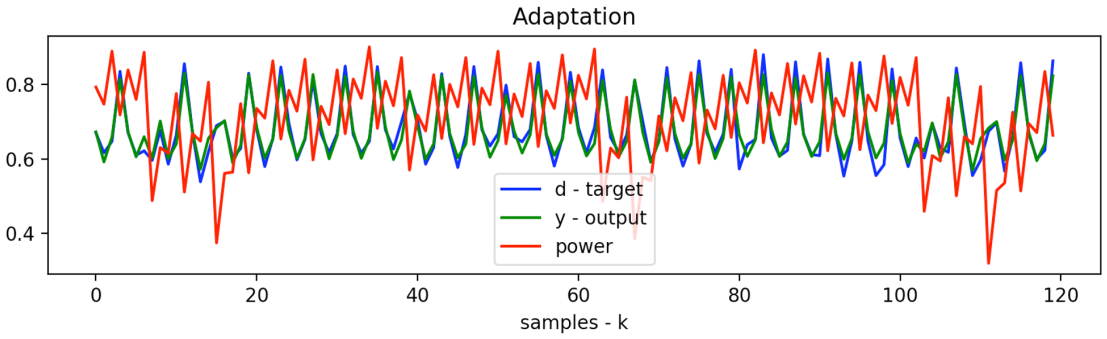


Fig. 2.15. Training on data having all possible key combinations and validating on any random key value

The training and validation data are the same as mentioned in experiment 4. Moreover, the MSE value while testing was observed to be 0.0095. Hence, we can now claim that we have successfully trained a model by considering all the parameters evenly.

2.2.9 Experiment 9

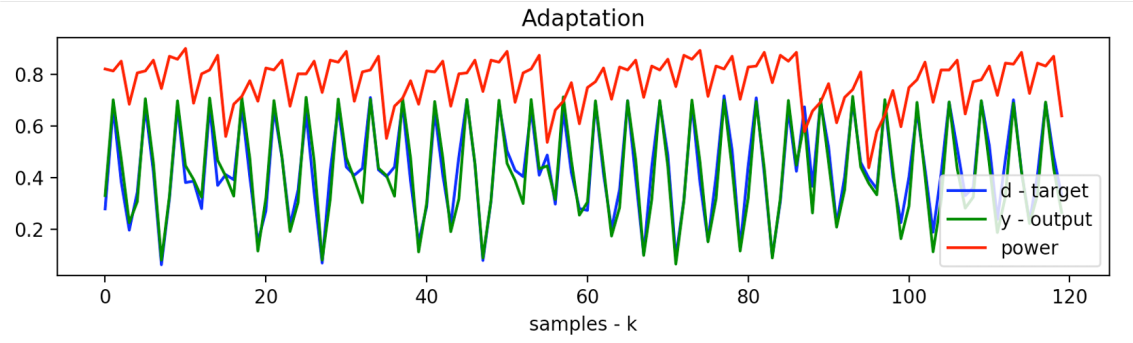


Fig. 2.16. Training on data having random plain text and key value while validating on data having a different random plain text and key value

The training and validation data are the same as mentioned in experiment 4. Moreover, the MSE value while testing was observed to be 0.0097. However, in experiments 8 and 9 given above, we observe a drastic change in the error value when we do not apply any data restrictions. Hence, even though the results are better than a filter we still need to fine-tune the network for validating our proposed leakage conversion technique. Finally, to summarize in this chapter we perform a comparative analysis between adaptive LMS FIR filters and ANNs. We also claim that ANNs are a better tool for performing leakage conversion. In the next chapter, we will be looking into the modifications made to the ANN for successfully running CEMA and a few other observations along the way.

3. LEAKAGE CONVERSION

If we want to claim that ANNs are a suitable tool for performing the conversion then we need to remove all the constraints we initially imposed on the input data and work with data over any time interval (Let's take 3000 time samples as we cannot test till infinity). Hence, we now look at the following graphical and heat map representations of our results from the various experiments performed with ANNs. Moreover, it should be noted that device number two was used to collect all the traces.

3.1 Power to Average EM

Figures 3.1 and 3.2 given below represents the output waveform from the trained neural network model (y-output) in green, the required EM trace waveform (d-target) in blue, and the input power trace waveform (power) in red. The network architecture used is as follows, the input layer consists of 3000 neurons for the 3000 time samples, the first hidden layer consists of 1028 neurons, the second hidden layer consists of 256 neurons, and the output layer again has 3000 neurons to generate the averaged EM trace having 3000 time samples. Moreover, the model is sequential with all linear layers and no dropout values. The hidden layers have sigmoid activation. For training 1000 power traces with 3000 time samples and 10000 EM traces (averaged by 10 to finally get 1000) with 3000 time samples were utilized. For testing 1500 power traces with 3000 time samples and 15000 EM traces (averaged by 10 to finally get 1500) with 3000 time samples were used. Moreover, the averaging was performed according to the key and plain text values. Other hyper-parameters include a learning rate of 0.001, 20 epochs, MSE as the cost function, and stochastic gradient descent as the optimizer.

3.1.1 Graphical representation of the result after using the modified network architecture

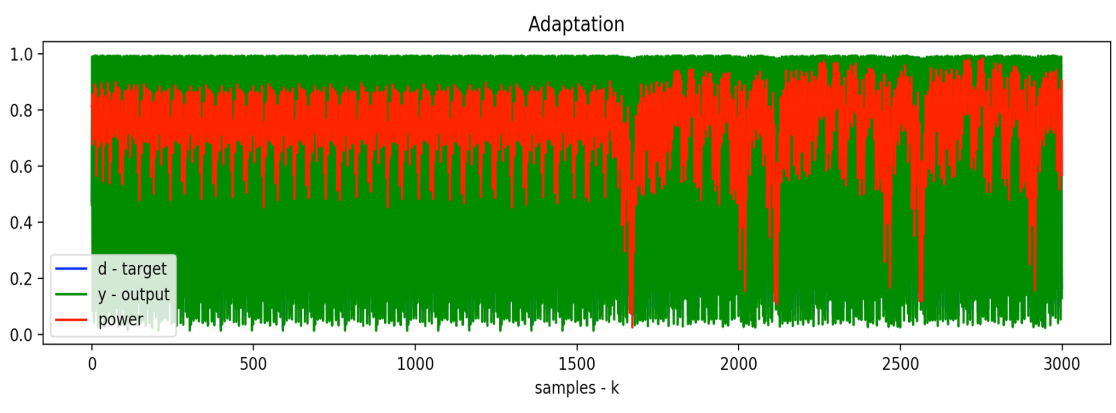


Fig. 3.1. Result of one trace and 3000 time samples

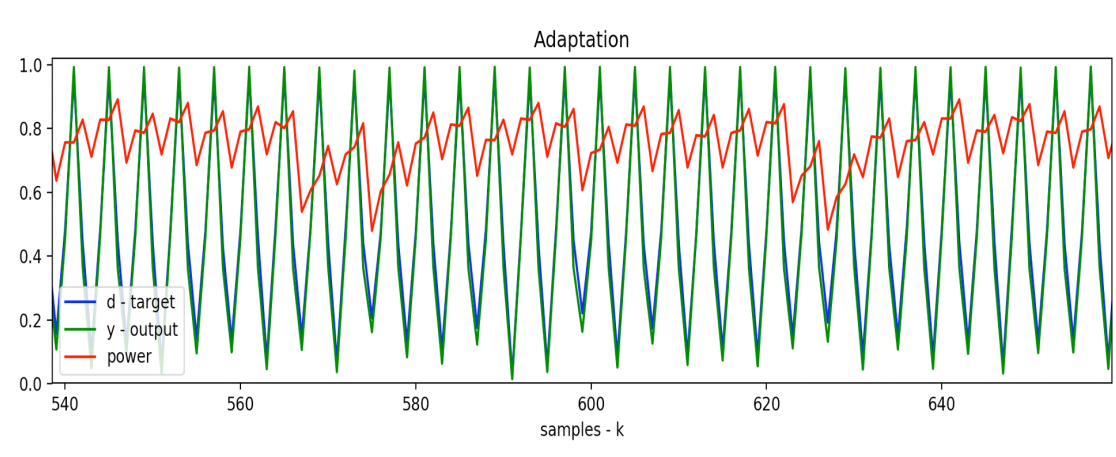


Fig. 3.2. Zoomed in sample output after training and testing the model

The MSE value while testing was observed to be 0.0262, which is quite low. Moreover, as we are now testing over a larger time domain the error must also be larger than the MSE values observed in chapter 2. However, from the above two graphs, we only observe one trace so with the help of a heat map we observe the MSE values by varying the keys and plain text values.

3.1.2 Heat Map

Note that in key 38 we have fixed a certain value for plain texts 21 to 40 so that we can observe what happens to the MSE values in that case.

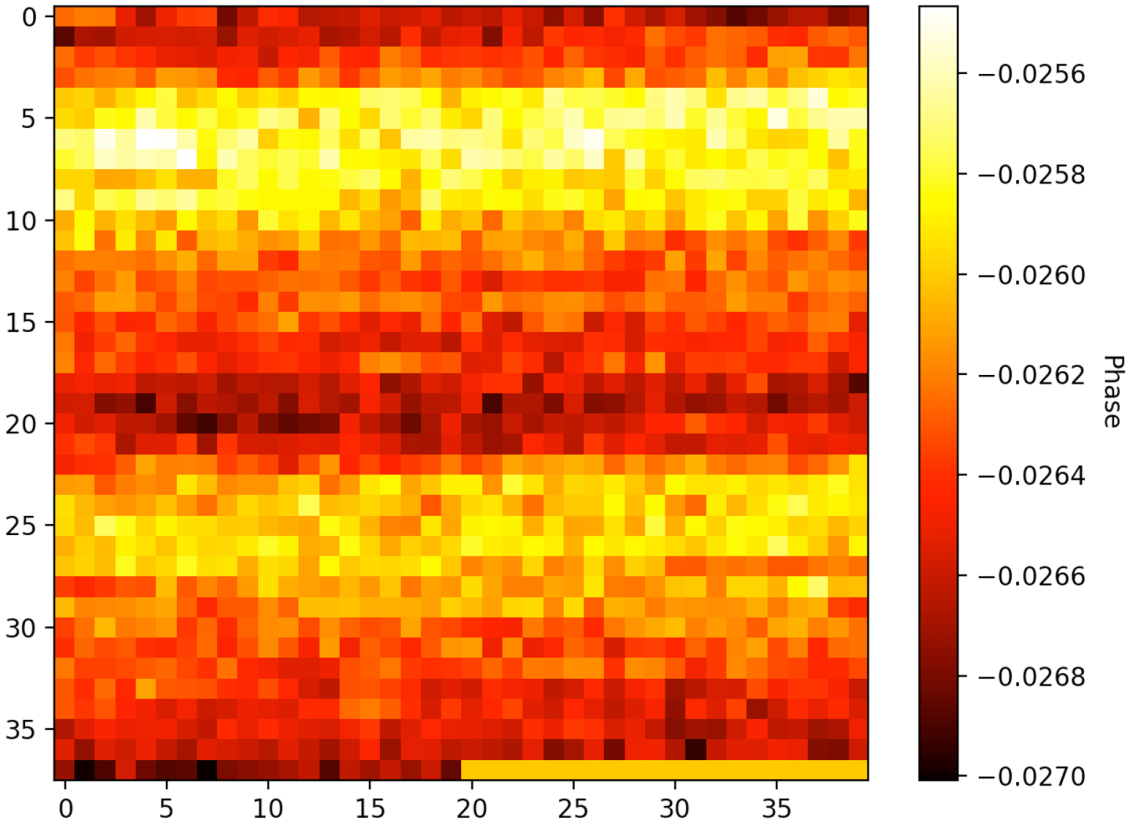


Fig. 3.3. Power to EM Heat Map for 38 different key values (rows) and 40 different plain texts (columns)

3.2 EM to Power

In this experiment, we try to inverse the object by generating power traces from raw EM traces. It is similar to performing an inverse of the complex function used to generate the EM from power traces. We observe the MSE values using the heat map below. Due to the high SNR characteristic of power traces we see a relatively low MSE value in each case. However, this is a much harder problem as we are trying

to remove noise from data already have a low SNR value. We also observe that CPA fails then the generated power traces are used. Hence, we can now claim that the inverse of the power to EM complex function does not exist by using ANNs. Again it should be noted that in key 38 we have fixed a certain value for plain texts 21 to 40 so that we can observe what happens to the MSE values in that case.

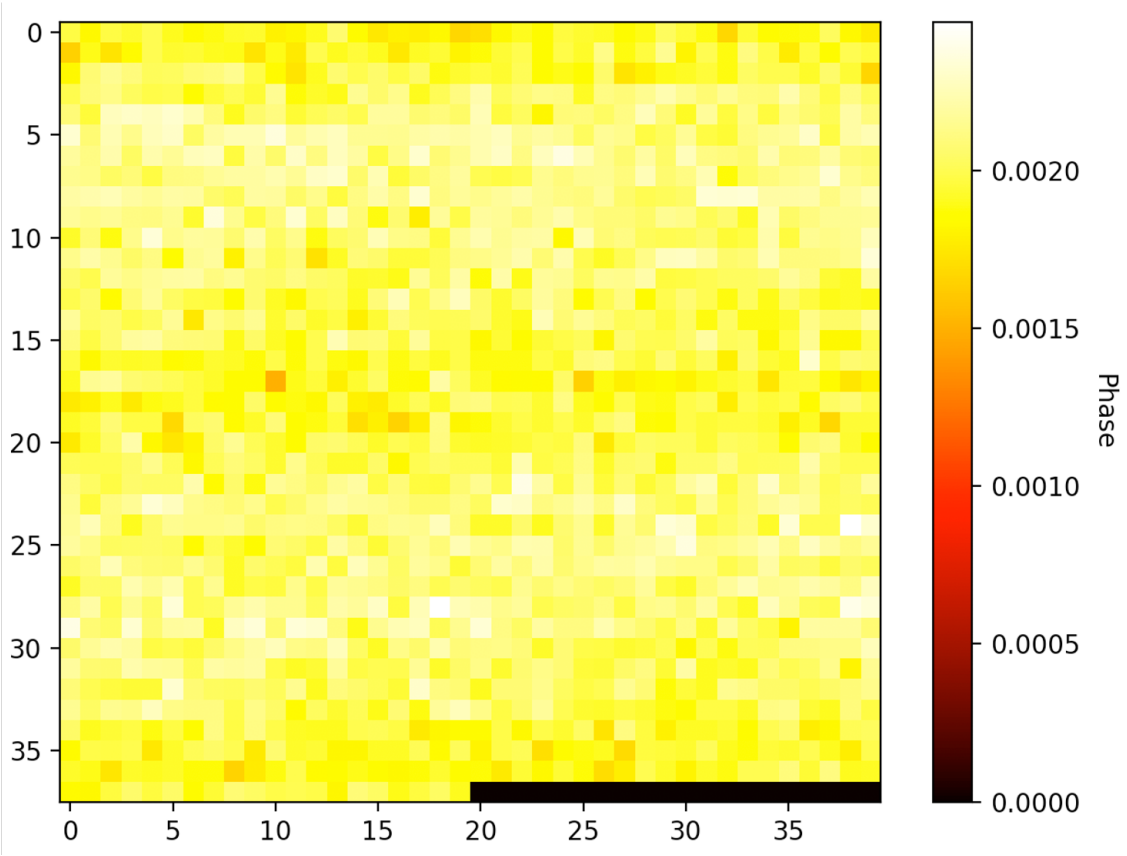


Fig. 3.4. EM to power heat map for 38 different key values (rows) and 40 different plain texts (columns)

3.3 CEMA

Now we need to make sure that the generated averaged EM traces can be used to extract the secret key. Hence, we use the common method of CEMA on the generated traces. Below are the graphical representations of our results. Here, we see

that for the majority of the keys (1,2, and 3) MTD is lesser than for raw traces but for key 0 MTD is almost the same as that of the raw traces. Moreover, for the graphs given below the red curve represents the correct key value and the remaining green ones represent all the other incorrect key values. Finally, with these curves, we can confidently claim the successful generation of the averaged EM traces.

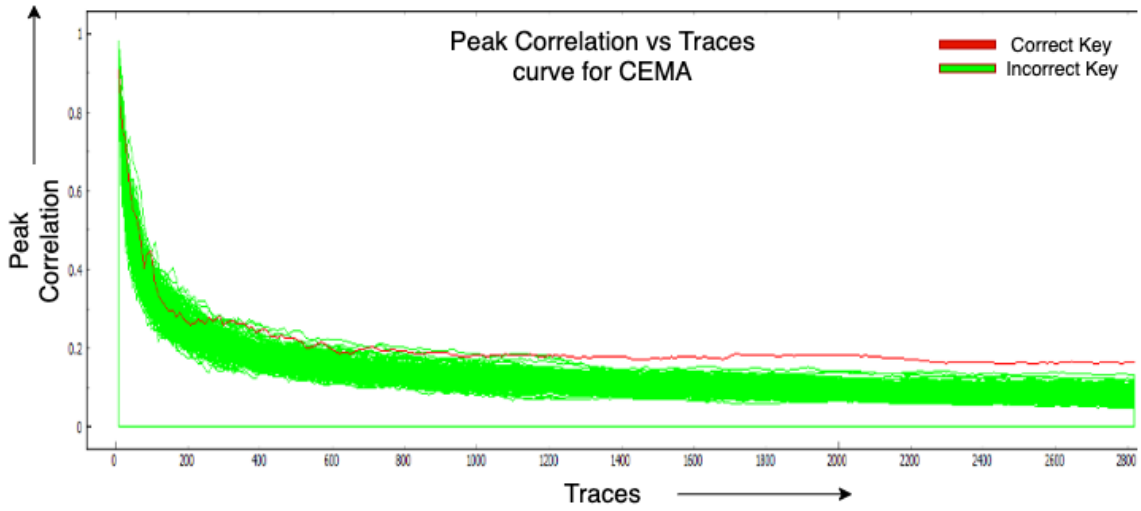


Fig. 3.5. Plot showing MTD for Key0 byte

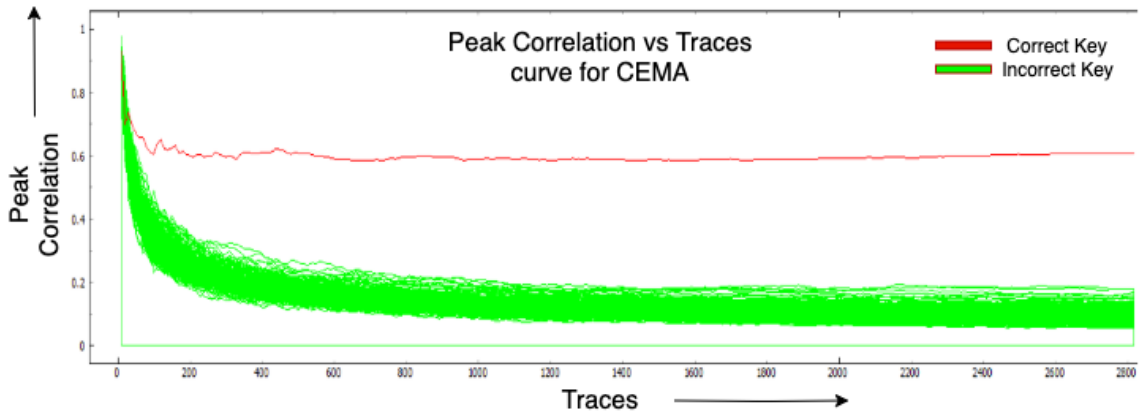


Fig. 3.6. Plot showing MTD for Key1 byte

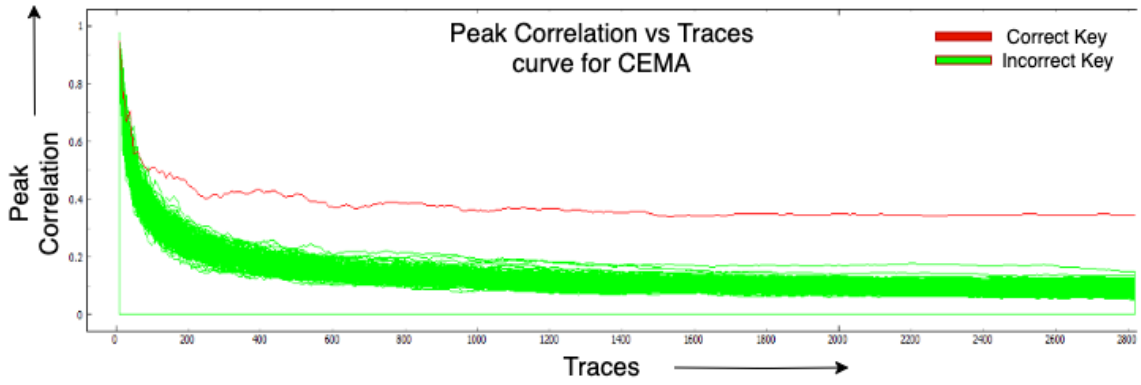


Fig. 3.7. Plot showing MTD for Key2 byte

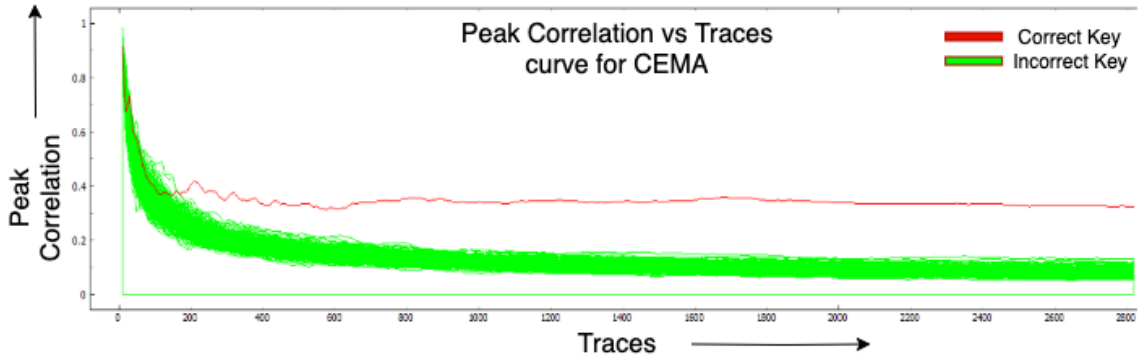


Fig. 3.8. Plot showing MTD for Key3 byte

3.4 N-Average Comparison

We perform this experiment to observe how the training and validation accuracy changes with different N-averaging values. Table 3.1 given below lists the number of traces used for different N values of averaging. Moreover, we need to observe these results to claim that there are some benefits of averaging.

3.4.1 Impact of averaging on training

From figure 3.9 given below, we observe that the training accuracy becomes constant after n=3 for the generated power traces but there is always some room for

Table 3.1.
Traces used for different N values of averaging

N value	**Training traces**	**Validation traces**
1	7987	998
3	4915	614
5	4300	538
7	4300	538
9	4096	512
11	4096	512
13	3891	486
15	3891	486
17	3891	486
19	3891	486
21	3686	461
23	3276	410
25	3072	384
27	2867	358
29	2662	333

improvement when generating the EM traces. From this curve, we can also infer why the inverse of the complex function when converting power to averaged EM was unsuccessful.

3.4.2 Impact of averaging on validation

From figure 3.10 given below, we observe that for the generated power traces validation accuracy initially increases till n=3 but degrades beyond it. However, for the generated EM traces validation accuracy increases till n=19 but beyond that, the

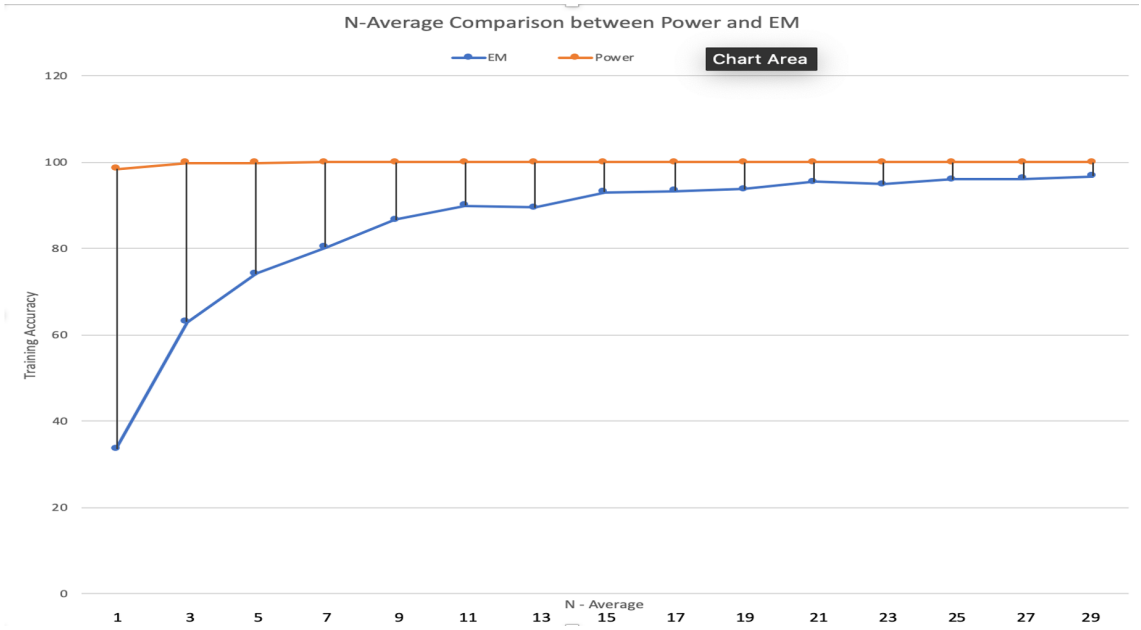


Fig. 3.9. Comparing using training accuracy

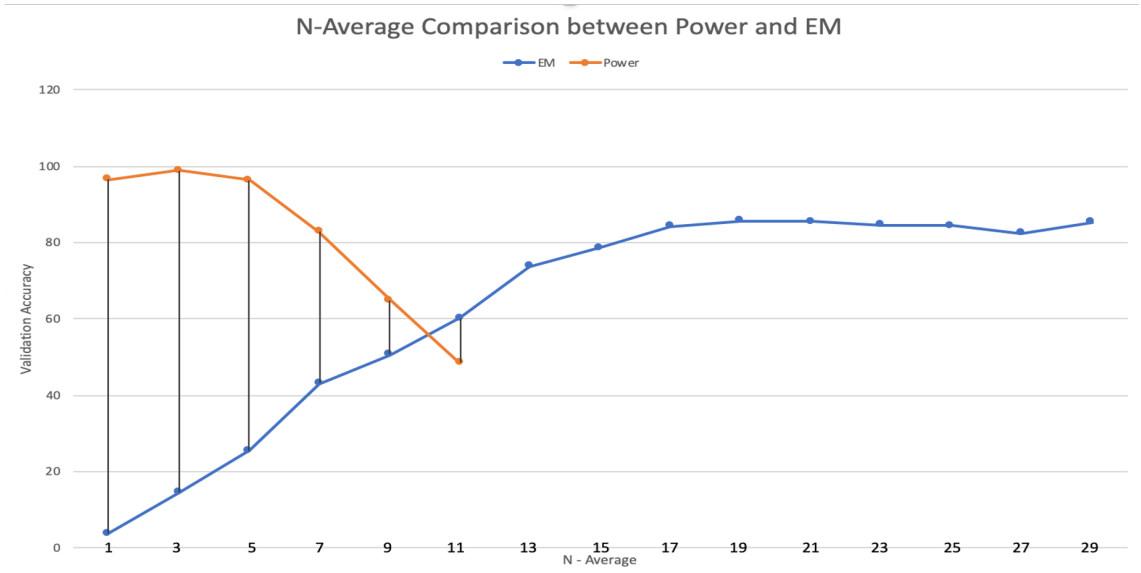


Fig. 3.10. Comparing using validation accuracy

curve flattens out. To summarize we can now claim that we do see an increase in validation and training accuracy with an increase in N values of averaging.

3.5 Modified Model for Reducing Error

Last time we observed an increase in MSE values for the heat map. Hence, in this experiment, we try to improve our model so that we can produce more realistic averaged traces.

3.5.1 Heat maps for converting power to averaged EM and EM to power by using the modified model

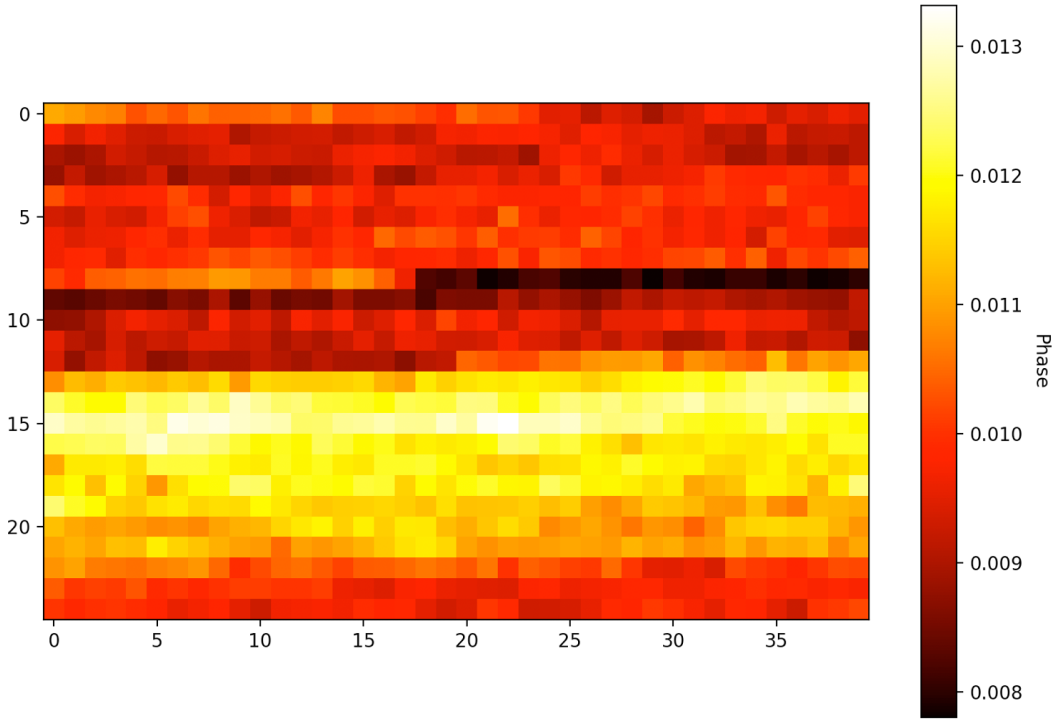


Fig. 3.11. Power to EM heat map for 25 same key values (rows) and 40 same plain texts (columns)

The hyper-parameters include a learning rate of 0.01, 3 epochs, MSE as the cost function, and stochastic gradient descent as the optimizer. The network architecture used is as follows, the input layer consists of 3000 neurons for the 3000 time samples, the first hidden layer consists of 2056 neurons, the second hidden layer consists of

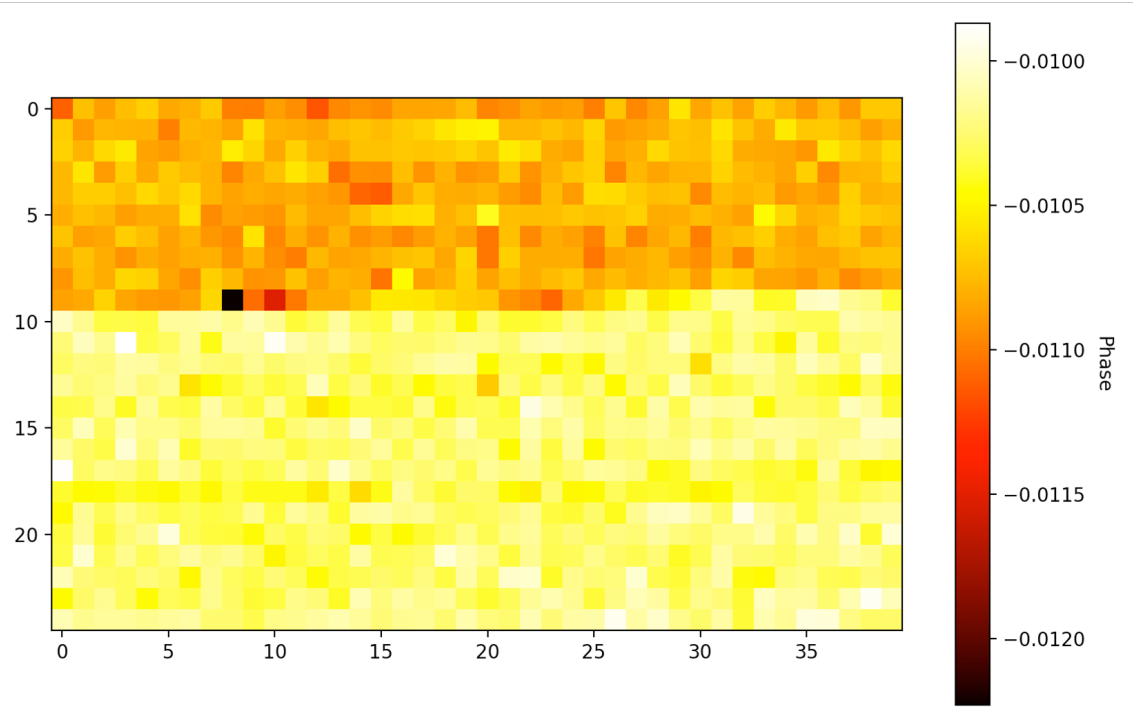


Fig. 3.12. EM to power heat map for 25 same key values (rows) and 40 same plain texts (columns)

1028 neurons, the third hidden layer consists of 512 neurons, the fourth hidden layer consists of 256 neurons, and the output layer again has 3000 neurons to generate the averaged EM trace having 3000 time samples. Moreover, the model is sequential with all linear layers and no dropout values. All the hidden layers have sigmoid activation. For training 1500 power traces with 3000 time samples and 15000 EM traces (averaged by 10 to finally get 1500) with 3000 time samples were utilized. For testing 1000 power traces with 3000 time samples and 10000 EM traces (averaged by 10 to finally get 1000) with 3000 time samples were used.

We use a fully connected neural network model. Sometimes an error can be higher when training and validating the data with the same key value. The reason for such an anomaly could be due to over-fitting. From the above results, we can state that the neural network learns the correlation between the power and EM traces. It does not learn as per the key values. The heat map shows the MSE for all 1500 traces

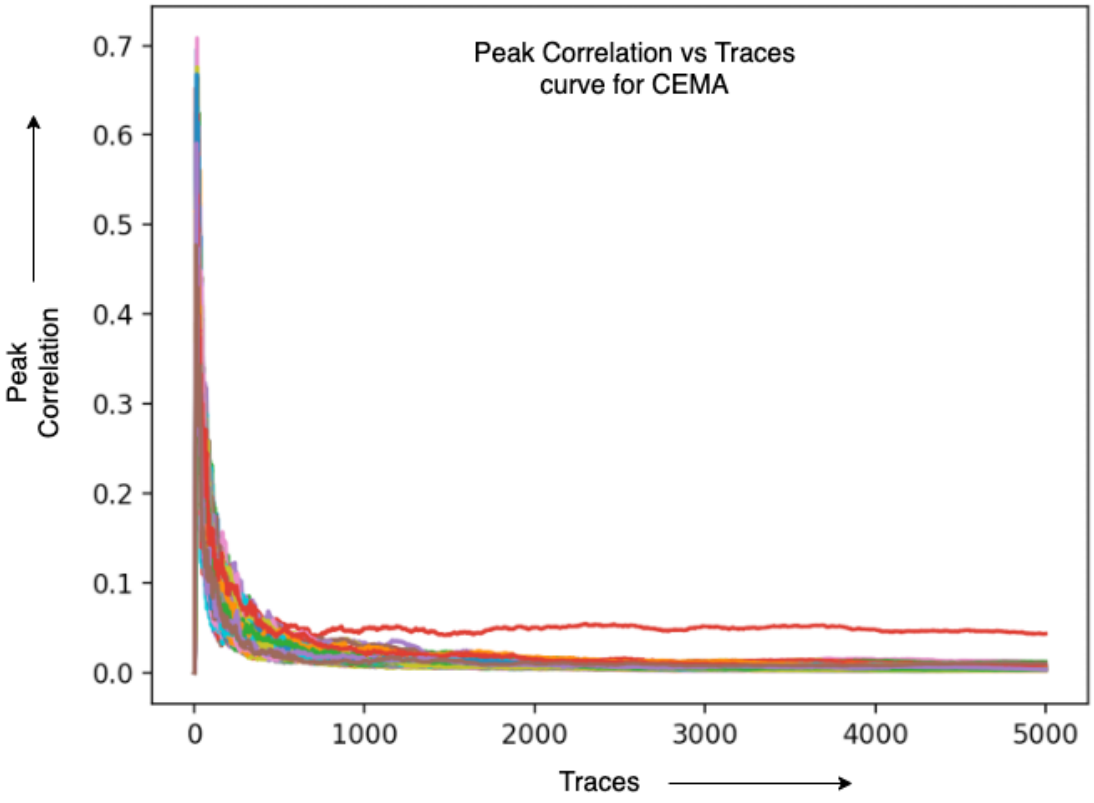


Fig. 3.13. Generated EM traces are tested for predicting the correct key value by using the modified model. The red curve which deviates out represents the correct key value and the rest represents the incorrect key value.

collected from device number two. Figure 3.13 portrays an MTD of 480 for key 0, which is a big improvement over the last model. Moreover, it also shows that the generated EM traces can predict the correct key values.

Finally, to summarize very briefly power to EM conversion gives an accuracy over 99% and the key can be predicted using CEMA but fails in the EM-ML-SCA model due to the minute errors the EM-ML-SCA model thinks of the traces from another device. EM to Power conversion gives an accuracy of over 99% but the key cannot be predicted using the CPA algorithm as during denoising the model removes the required information from the traces. As the conversion function will be a higher degree polynomial which cannot produce 100% accuracy.

4. SUMMARY AND CONCLUSION

In this thesis, we focus and try to improve the training stage of machine learning-based EM SCA. Recently, these ML-based SCA models have shown to be very efficient in extracting the secret key but they require a lot of data for training. Hence, we utilize the new technique of leakage conversion, where we convert power traces to N-averaged EM traces. Initially, we used adaptive DSP algorithms for the conversion but unfortunately, the generated traces were no way similar to the required traces. Next, we utilized neural networks for learning the non-linear dependencies which the adaptive DSP algorithm was unable to learn. Now, fortunately, the generated traces from the ANN model have very little error and are almost similar to the actual traces. Thus, we can claim that neural networks are better than DSP algorithms in this scenario. Moreover, we show the successful conversion of traces having a random key and plain text values. Finally, to further validate our generated traces we successfully run CEMA on our generated traces, where we observe the correct key value to diverge away from the rest. Hence, correctly validating the generated traces and completing the leakage conversion technique. A few additional works done in this thesis include showing how EM to power conversion is a much harder problem than the power to EM and how averaging impacts the training and validation accuracy. To conclude in this thesis we successfully convert power to averaged EM traces. Now for future work and our recommendation for training ML SCA models would be to adopt a key specific leakage conversion approach. As noise is random for each key-value we suggest training a separate model for each key so that the generated traces resemble more like the actual one. Moreover, in this thesis, we implemented this method which currently gives a low validation accuracy. Thus validating our approach of leakage conversion for training ML SCA models faster.

5. RECOMMENDATIONS AND FUTURE WORK

We aim to speed up the training process of the EM-ML-SCA model by converting the power traces to averaged EM traces. For training EM-ML-SCA model by 20,000 traces we need to collect 200,000 EM traces then again average by 10. But by converting power to averaged EM we only need 20,000 power traces with the corresponding key values which give a 10x improvement in training speed. Hence, for the recommendation, we propose a new model which we have already tested. It produces better results than the models mentioned in the previous chapters. The details are as follows. Moreover, for future work, we propose a key specific model training.

25600 power traces from device 2 was collected. 51200 averaged EM traces (averaged by 10) from device 2 was collected. This 51200 averaged EM traces were divided equally into two parts. The second part having 25600 averaged EM traces were used for training and validating the EM-ML-SCA model. In the EM-ML-SCA model after performing principal component analysis, 20482 traces were used in training and 2560 traces were used to validate. Validation accuracy is 78.75% and loss equals 0.672491542622447. Note that the EM-ML-SCA model uses only plain text 0 and predicts key byte 0.

The 25600 power traces and the first half of the averaged EM traces (25600) were used in training and validation. Learning rate equals 0.01 and epochs equals 5. The input layer has 3000 neurons, the second layer has 1000 neurons, the third layer has 500 neurons and finally, the output layer has 3000 neurons. Both hidden layers are of Relu type. Adam optimizer is used instead of Sigmoid. Originally The model is trained separately for 256 different key values. For each key-value, there are 100 traces. Thus, 80 traces are used for training and 20 are used for validation.

From figure 5.6 we propose that the benefit will come once all the 256 models are trained; we can reuse that model to generate more EM traces thus reducing the

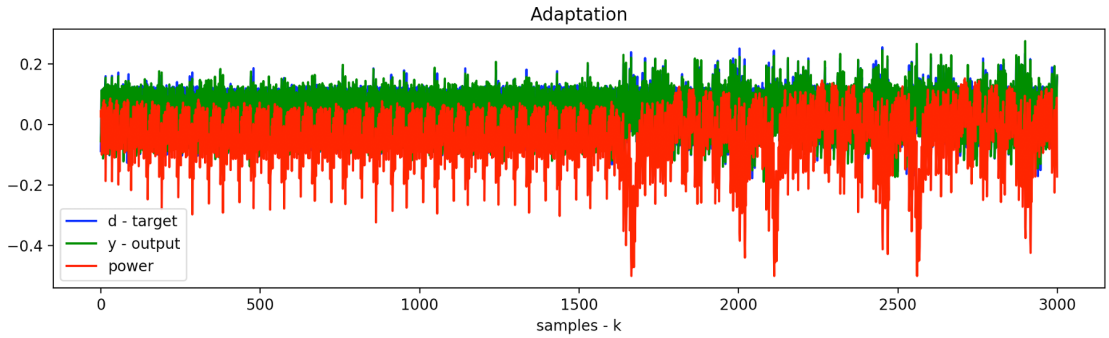


Fig. 5.1. Trace 71 with error 0.0007

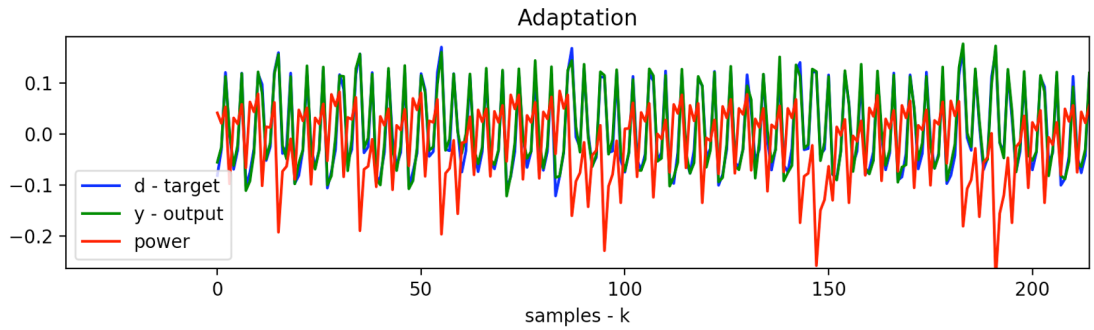


Fig. 5.2. Zoomed in trace 71 with error 0.0007

collection time from the physical device. We generally want to train with more traces to achieve greater validation accuracy so this model will come in handy for reducing the time required for collecting traces from the physical device.

Figures 5.1 and 5.2 represents the output waveform from the above mentioned trained neural network model (y-output) in green, the required EM trace waveform (d-target) in blue, and the input power trace waveform (power) in red. Finally, figures 5.3 to 5.6 represent very preliminary results to show how the future work needs to carry on after the successful implementation of the leakage conversion method. Moreover, in figure 5.5 we train on 70 traces belonging to the first half of the EM trace dataset and validate on all 100 original EM traces from the second half of the dataset. Another idea for future work can be to create a custom loss function [43]

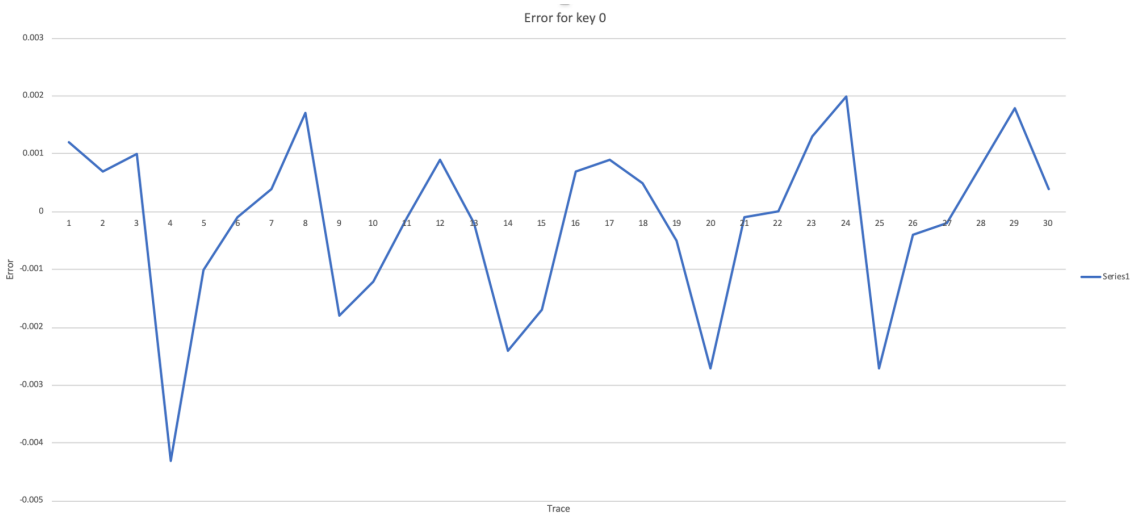


Fig. 5.3. Validating the 30 generated traces for key0

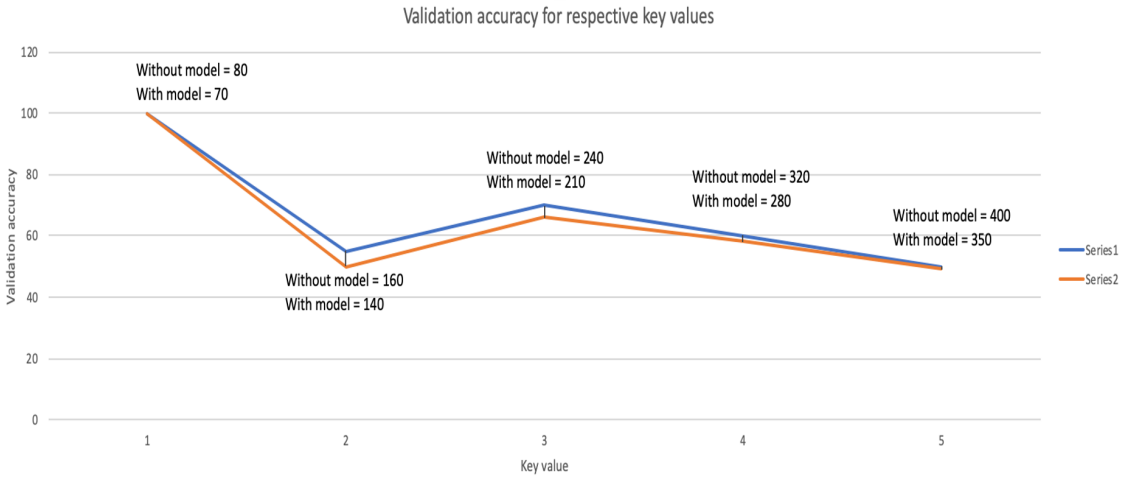


Fig. 5.4. Validation accuracy for respective key values

which will simultaneously find the time interval for the encryption operation and learn the transformation at that instance [44]. This should prevent the model from learning unnecessary noise [45].

```
24/24 [==============================] - 1s 30ms/sample - loss: 7.3011 - acc: 0.0000e+00 - val_loss: 5.2336 - val_acc: 1.0000
Epoch 2/15
24/24 [==============================] - 0s 2ms/sample - loss: 6.3702 - acc: 0.0000e+00 - val_loss: 4.9159 - val_acc: 1.0000
Epoch 3/15
24/24 [==============================] - 0s 2ms/sample - loss: 6.0007 - acc: 0.0417 - val_loss: 4.6790 - val_acc: 1.0000
Epoch 4/15
24/24 [==============================] - 0s 2ms/sample - loss: 6.0099 - acc: 0.1667 - val_loss: 4.5239 - val_acc: 1.0000
Epoch 5/15
24/24 [==============================] - 0s 2ms/sample - loss: 5.8570 - acc: 0.0833 - val_loss: 4.4625 - val_acc: 1.0000
Epoch 6/15
24/24 [==============================] - 0s 3ms/sample - loss: 5.2624 - acc: 0.2083 - val_loss: 4.4321 - val_acc: 1.0000
Epoch 7/15
24/24 [==============================] - 0s 3ms/sample - loss: 4.7608 - acc: 0.3333 - val_loss: 4.3038 - val_acc: 1.0000
Epoch 8/15
24/24 [==============================] - 0s 2ms/sample - loss: 4.3785 - acc: 0.4167 - val_loss: 4.0848 - val_acc: 1.0000
Epoch 9/15
24/24 [==============================] - 0s 2ms/sample - loss: 3.7025 - acc: 0.3333 - val_loss: 3.7737 - val_acc: 1.0000
Epoch 10/15
24/24 [==============================] - 0s 2ms/sample - loss: 3.0325 - acc: 0.4167 - val_loss: 3.4091 - val_acc: 1.0000
Epoch 11/15
24/24 [==============================] - 0s 3ms/sample - loss: 2.7549 - acc: 0.6250 - val_loss: 3.0450 - val_acc: 1.0000
Epoch 12/15
24/24 [==============================] - 0s 2ms/sample - loss: 2.4736 - acc: 0.6250 - val_loss: 2.6661 - val_acc: 1.0000
Epoch 13/15
24/24 [==============================] - 0s 2ms/sample - loss: 1.7638 - acc: 0.6250 - val_loss: 2.0607 - val_acc: 1.0000
Epoch 14/15
24/24 [==============================] - 0s 2ms/sample - loss: 1.1547 - acc: 0.8333 - val_loss: 1.3849 - val_acc: 1.0000
Epoch 15/15
24/24 [==============================] - 0s 3ms/sample - loss: 1.0523 - acc: 0.8333 - val_loss: 0.7138 - val_acc: 1.0000
3/3 [==============================] - 0s 927us/sample - loss: 0.7171 - acc: 1.0000
[0.7170610427856445, 1.0]
100/100 [==============================] - 0s 428us/sample - loss: 1.0681 - acc: 1.0000
[1.068100280761718, 1.0]
```

Fig. 5.5. Some preliminary results on implementing model specific training

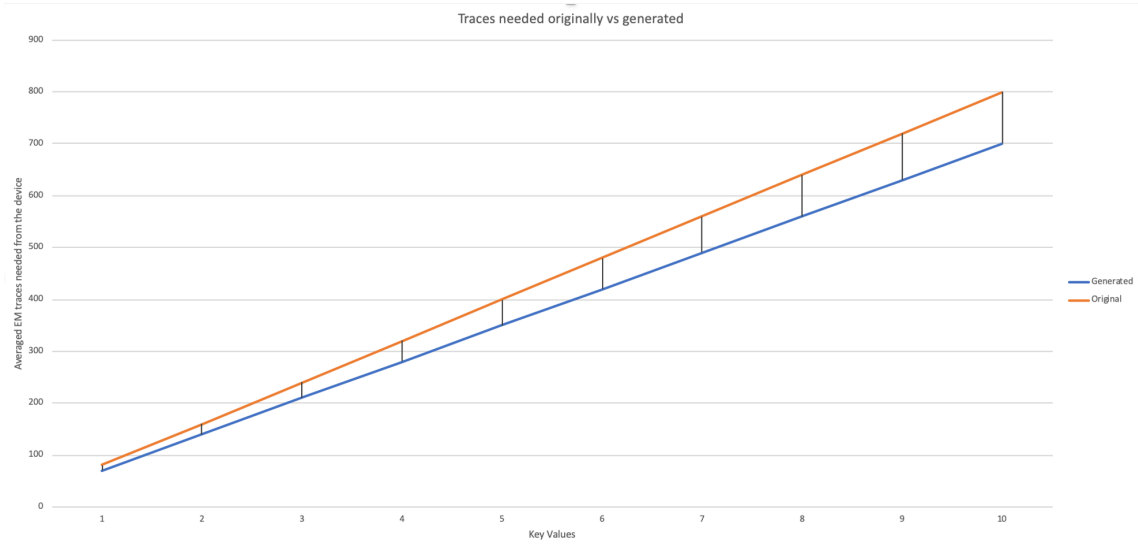


Fig. 5.6. Benefit of the model

REFERENCES

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[2] E. Brier, C. Clavier, and F. Olivier, "Optimal statistical power analysis," Cryptology ePrint Archive, Report 2003/152, 2003, https://eprint.iacr.org/2003/152.

[3] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Smart Card Programming and Security*, I. Attali and T. Jensen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 200–210.

[4] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 251–261.

[5] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology — CRYPTO '96*, N. Koblitz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 104–113.

[6] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Computer Networks*, vol. 48, no. 5, pp. 701 – 716, 2005, web Security. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128605000125

[7] D. Genkin, A. Shamir, and E. Tromer, "Rsa key extraction via low-bandwidth acoustic cryptanalysis," in *Advances in Cryptology – CRYPTO 2014*, J. A. Garay and R. Gennaro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 444–461.

[8] H. Kour and R. K. Jha, "Em radiation reduction in wcn: Towards safe generations," in *2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*, Jan 2020, pp. 559–562.

[9] L. Onsager, "Deviations from ohm's law in weak electrolytes," *The Journal of Chemical Physics*, vol. 2, no. 9, pp. 599–615, 1934. [Online]. Available: https://doi.org/10.1063/1.1749541

[10] D. Oswald, B. Richter, and C. Paar, "Side-channel attacks on the yubikey 2 one-time password generator," in *Research in Attacks, Intrusions, and Defenses*, S. J. Stolfo, A. Stavrou, and C. V. Wright, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 204–222.

[11] D. Oswald and C. Paar, "Breaking mifare desfire mf3icd40: Power analysis and templates in the real world," in *Cryptographic Hardware and Embedded Systems – CHES 2011*, B. Preneel and T. Takagi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 207–222.

[12] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, B. S. Kaliski, ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 13–28.

[13] C. Clavier, D. Marion, and A. Wurcker, "Simple power analysis on aes key expansion revisited," in *Cryptographic Hardware and Embedded Systems – CHES 2014*, L. Batina and M. Robshaw, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 279–297.

[14] E. De Mulder, P. Buysschaert, S. B. Ors, P. Delmotte, B. Preneel, G. Vandenbosch, and I. Verbauwhede, "Electromagnetic analysis attack on an fpga implementation of an elliptic curve cryptosystem," in *EUROCON 2005 - The International Conference on "Computer as a Tool"*, vol. 2, Nov 2005, pp. 1879–1882.

[15] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 16–29.

[16] T.-H. Le, J. Clédière, C. Canovas, B. Robisson, C. Servière, and J.-L. Lacoume, "A proposition for correlation power analysis enhancement," in *Cryptographic Hardware and Embedded Systems - CHES 2006*, L. Goubin and M. Matsui, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 174–186.

[17] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, "X-deepsca: Cross-device deep learning side channel attack," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3316781.3317934

[18] D. Das, S. Maity, S. B. Nasir, S. Ghosh, A. Raychowdhury, and S. Sen, "Asni: Attenuated signature noise injection for low-overhead power side-channel attack immunity," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 10, pp. 3300–3311, Oct 2018.

[19] ——, "High efficiency power side-channel attack immunity using noise injection in attenuated signature domain," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, May 2017, pp. 62–67.

[20] D. Das, M. Nath, B. Chatterjee, S. Ghosh, and S. Sen, "Stellar: A generic em side-channel attack protection through ground-up root-cause analysis," *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2019.

[21] A. Peled and B. Liu, *Digital signal processing: Theory, design, and implementation.* New York, John Wiley and Sons, Inc., 1976.

[22] A. V. Oppenheim, *Applications of digital signal processing.* Englewood Cliffs, N.J., Prentice-Hall, Inc., 1978.

[23] Xiangkun Chen and T. Parks, "Design of fir filters in the complex domain," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 2, pp. 144–153, February 1987.

[24] Sunghwan Ong, Cheolwoo You, Sooyong Choi, and Daesik Hong, "A decision feedback recurrent neural equalizer as an infinite impulse response filter," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2851–2858, Nov 1997.

[25] M. G. Bellanger, *Adaptive digital filters*. Marcel Dekker, 2001.

[26] M. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14, pp. 2627 – 2636, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1352231097004470

[27] R. C. Elson, A. I. Selverston, R. Huerta, N. F. Rulkov, M. I. Rabinovich, and H. D. I. Abarbanel, "Synchronous behavior of two coupled biological neurons," *Phys. Rev. Lett.*, vol. 81, pp. 5692–5695, Dec 1998. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.81.5692

[28] M. Ghiassi, H. Saidane, and D. Zimbra, "A dynamic artificial neural network model for forecasting time series events," *International Journal of Forecasting*, vol. 21, no. 2, pp. 341 – 362, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207004001116

[29] M. A. Cowley, J. L. Smart, M. Rubinstein, M. G. Cerdán, S. Diano, T. L. Horvath, R. D. Cone, and M. J. Low, "Leptin activates anorexigenic pomc neurons through a neural network in the arcuate nucleus," *Nature*, vol. 411, no. 6836, pp. 480–484, 2001. [Online]. Available: https://doi.org/10.1038/35078085

[30] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, "Report on the development of the advanced encryption standard (aes)," *Journal of research of the National Institute of Standards and Technology*, vol. 106, no. 3, pp. 511–577, Jun 2001. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/27500035

[31] L. Lerman, R. Poussier, O. Markowitch, and F.-X. Standaert, "Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version," *Journal of Cryptographic Engineering*, vol. 8, no. 4, pp. 301–313, 2018. [Online]. Available: https://doi.org/10.1007/s13389-017-0162-9

[32] P. Kubáň and P. C. Hauser, "Fundamental aspects of contactless conductivity detection for capillary electrophoresis. part ii: Signal-to-noise ratio and stray capacitance," *ELECTROPHORESIS*, vol. 25, no. 20, pp. 3398–3405, 2004. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/elps.200406060

[33] S. C. Douglas, Quanhong Zhu, and K. F. Smith, "A pipelined lms adaptive fir filter architecture without adaptation delay," *IEEE Transactions on Signal Processing*, vol. 46, no. 3, pp. 775–779, March 1998.

[34] F. Reed, P. Feintuch, and N. Bershad, "Time delay estimation using the lms adaptive filter–static behavior," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 3, pp. 561–571, June 1981.

[35] R. E. Uhrig, "Introduction to artificial neural networks," in *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, vol. 1, Nov 1995, pp. 33–37 vol.1.

[36] J. W. Halle, A. M. Marshall, and J. E. Spradlin, "Time delay: A technique to increase language use and facilitate generalization in retarded children," *Journal of Applied Behavior Analysis*, vol. 12, no. 3, pp. 431–439, 1979. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1901/jaba.1979.12-431

[37] B. Thirion, G. Flandin, P. Pinel, A. Roche, P. Ciuciu, and J.-B. Poline, "Dealing with the shortcomings of spatial normalization: Multi-subject parcellation of fmri datasets," *Human Brain Mapping*, vol. 27, no. 8, pp. 678–693, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/hbm.20210

[38] M. Rezaeian Zadeh, S. Amin, D. Khalili, and V. P. Singh, "Daily outflow prediction by multi layer perceptron with logistic sigmoid and tangent sigmoid activation functions," *Water Resources Management*, vol. 24, no. 11, pp. 2673–2688, 2010. [Online]. Available: https://doi.org/10.1007/s11269-009-9573-4

[39] C. Chinrungrueng and C. H. Sequin, "Optimal adaptive k-means algorithm with dynamic adjustment of learning rate," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 157–169, Jan 1995.

[40] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005. [Online]. Available: https://www.int-res.com/abstracts/cr/v30/n1/p79-82/

[41] S. F. Crone, "Training artificial neural networks for time series prediction using asymmetric cost functions," in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02.*, vol. 5, Nov 2002, pp. 2374–2380 vol.5.

[42] S. Ruder, "An overview of gradient descent optimization algorithms," 2016.

[43] C. G. Rowbottom, S. Webb, and M. Oldham, "Beam-orientation customization using an artificial neural network," *Physics in Medicine and Biology*, vol. 44, no. 9, pp. 2251–2262, aug 1999. [Online]. Available: https://doi.org/10.1088%2F0031-9155%2F44%2F9%2F312

[44] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural Computation*, vol. 4, no. 4, pp. 473–493, 1992. [Online]. Available: https://doi.org/10.1162/neco.1992.4.4.473

[45] A. Rahideh and M. H. Shaheed, "Mathematical dynamic modelling of a twin-rotor multiple input-multiple output system," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 221, no. 1, pp. 89–101, 2007. [Online]. Available: https://doi.org/10.1243/09596518JSCE292

VITA

VITA

Rohan is currently a Master's student in the Electrical and Computer Engineering department and a Graduate Teaching Assistant in the Computer Science department at Purdue University. He serves as a lab lead and has taught object-oriented programming to over 200 students over the past two years. His master's thesis focuses on the topic of hardware security, proving artificial neural networks give better accuracy than traditional digital signal processing algorithms. He will continue to pursue a Ph.D. degree from Purdue University, where he will continue his work on Machine/Deep Learning algorithms. Before coming to Purdue, he completed his Bachelor's degree from KIIT University, India. During his undergraduate studies, he worked on image processing and software cost estimation using a modified particle swamp optimization algorithm. He anticipates developing robust machine learning algorithms for autonomous racing vehicles.