IMPROVED STUDY OF SIDE-CHANNEL ATTACKS USING RECURRENT NEURAL NETWORKS

by

Muhammad Abu Naser Rony Chowdhury

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

December 2019

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Muhammad Abu Naser Rony Chowdhury

Thesis Title: Improved Study of Side-Channel Attacks Using Recurrent Neural Networks

Date of Final Oral Examination: 19 August 2019

The following individuals read and discussed the thesis submitted by student Muhammad Abu Naser Rony Chowdhury, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

John Stubban, Ph.D.	Chair, Supervisory Committee
Marion Scheepers, Ph.D.	Member, Supervisory Committee
Casey Kennington, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by John Stubban, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

To my parents and my wife, who always believe in me.

ACKNOWLEDGMENTS

First I would like to express my gratitude to the Almighty for allowing me to finish my work. I cannot express enough thanks to my committee for their continued support and encouragement: Dr. John Stubban, my major professor; Dr. Marion Scheepers and Dr. Casey Kennington, my honorable committee members. I offer my sincere appreciation for the learning opportunities provided by my committee. I would like to express my heartiest thanks to Dr. John Stubban for going extra miles by scrutinizing my works and making it look solid. A special thanks to Dr. Yantian Hou and Dr. Gaby Dagher for providing important insights about my work. My completion of this thesis work could not have been accomplished without the support of my wife, Emu. To Emu - thank you for allowing me time away from you to research and write. Thanks to my parents as well, Mr. and Mrs. Chowdhury, for their love and blessings. Last, but not least, I would like to thank Ahnaf and Ahiyan, my two sons, who have been a continuous inspiration for my work.

I am very much thankful to the Computer Science Department for giving me the opportunity to flourish myself as a researcher by providing financial and advisory support. I would like to thank Dr. Jerry Alan Fails, graduate coordinator, Jordan Morales, and Susie Gillikin for their tremendous support and help towards me.

ABSTRACT

Differential power analysis attacks are special kinds of side-channel attacks where power traces are considered as the side-channel information to launch the attack. These attacks are threatening and significant security issues for modern cryptographic devices such as smart cards, and Point of Sale (POS) machines; because after careful analysis of the power traces, the attacker can break any secured encryption algorithm and can steal sensitive information.

In our work, we study differential power analysis attacks using two popular neural networks: Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). Our work seeks to answer three research questions(RQs):

RQ1: Is it possible to predict the unknown cryptographic algorithm using neural network models from different datasets?

RQ2: Is it possible to map the key value for the specific plaintext-ciphertext pair with or without side-band information?

RQ3: Using similar hyper-parameters, can we evaluate the performance of two neural network models (CNN vs. RNN)?

In answering these questions, we have worked with two different datasets: one is a physical dataset (DPA contest v1 dataset), and the other one is a simulated dataset (toggle count quantity) from Verilog HDL. We have evaluated the efficiency of CNN and RNN models in predicting the unknown cryptographic algorithms of the device under attack. We have mapped to 56 bits key for a specific plaintext-ciphertext pair with and without using side-band information. Finally, we have evaluated our neural network models using different metrics such as accuracy, loss, baselines, epochs, speed of operation, memory space consumed, and so on. We have shown the performance comparison between RNN and CNN on different datasets. We have done three experiments and shown our results on these three experiments. The first two experiments have shown the advantages of choosing CNN over RNN while working with side-channel datasets. In the third experiment, we have compared two RNN models on the same datasets but different dimensions of the datasets.

TABLE OF CONTENTS

ABSTRACT vi

LIST OF TABLES xi

LIST OF FIGURES xii

LIST OF ABBREVIATIONS xiv

1 Introduction 1

1.1 Thesis Statement 3

1.2 Our Major Contributions 4

1.3 Outline 5

2 Terminologies 7

2.1 Side-Channel Attack 7

2.2 Power Analysis Attack 8

2.2.1 Simple Power Analysis (SPA) 9

2.2.2 Differential Power Analysis Attack (DPA) 11

2.3 Deep Learning Techniques 13

2.3.1 Artificial Neural Network 13

2.3.2 Convolution Neural Network (CNN) 15

2.3.3 Recurrent Neural Network (RNN) 18

2.3.4 Long-Short Term Memory (LSTM) Network 19

2.3.5 Keras . . . 20
2.3.6 Tensorflow . . . 20
2.3.7 Data Encryption Standard . . . 20
2.3.8 Advanced Encryption Standard (AES) . . . 21
2.4 The DPA Contest Dataset . . . 22
2.4.1 DPA Contest V1: DES Dataset . . . 23
3 Related Work . . . 24
4 Methodology . . . 26
4.1 Big Picture of the thesis Work . . . 26
4.2 Datasets . . . 27
4.2.1 Dataset Structure . . . 27
4.2.2 Dataset Cleaning (Munging) and Visualization . . . 29
4.3 Deep Learning Experiments . . . 30
4.4 Experiment 1: Prediction of DES rounds . . . 30
4.4.1 RNN Model to Predict on DPA Contest Data . . . 31
4.4.2 RNN model on Toggle Count Datasets . . . 32
4.4.3 Baselines . . . 32
4.4.4 Result: Accuracy and Loss Evaluation in Predicting . . . 32
4.4.5 Discussion . . . 35
4.5 Experiment 2: Classification of Two Classes: DES and Non-DES . . . 36
4.5.1 Procedure of the Experiment . . . 36
4.5.2 Baselines . . . 37
4.5.3 Result Comparison of RNN and CNN in Classifying Two Classes 38
4.5.4 Discussion . . . 39

4.6 Experiment 3: Recurrent Neural Network Model for Mapping Keys . . . 40
4.6.1 Procedure of the Experiment 40
4.6.2 Baselines or Metrics 42
4.6.3 Performance Evaluation of RNN Model with and without Sideband Information 43
4.6.4 Discussion 44

5 Conclusion 68
5.1 Overview of Our Work and Contributions 68
5.2 Directions of Future Work 69

REFERENCES 70

A Correlation Power Analysis (CPA) 72
A.1 Overview 72
A.2 Key Guess using Python Program 72
A.3 Recurrent Neural Network 74

B Power Measurement Setup for Xilinx Artix-7 Board 76
B.1 Overview 76

LIST OF TABLES

4.1 Power Traces values are shown in the table 27

4.2 Information of DPA Contest Dataset 28

4.3 Toggle Count Datasets from Verilog 29

4.4 Comparison of CNN & RNN on different matrices 36

4.5 Comparison of CNN & RNN on Binary Classification 39

4.6 Comparison of RNN & RNN on Key Mapping 44

LIST OF FIGURES

2.1 A typical diagram of Side Channel Attack 8

2.2 A depiction of Power Analysis Attack 10

2.3 A single power trace of DES algorithm [19] 11

2.4 Differential Power Analysis Attack overview 12

2.5 A simple Artificial Neural Network Structure 16

2.6 Overview of Convolutional Neural Network Operation 17

2.7 A pictorial view of DES algorithm 22

2.8 A pictorial view of AES algorithm 23

4.1 Overview of the hierarchical pattern of the thesis work 45

4.2 Initial condition 46

4.3 After Cleaning 46

4.4 DPA contest dataset after applying windowing functions 46

4.5 The toggle count dataset pattern 47

4.6 DES rounds prediction with 1000 traces 48

4.7 DES rounds prediction with 5000 traces 49

4.8 DES rounds prediction with 50000 traces 50

4.9 DES rounds prediction with 350 sets of Toggle Count traces 51

4.10 RNN Model Accuracy on Toggle Quantity Count Dataset 52

4.11 RNN Model Loss on Toggle Quantity Count Dataset 53

4.12 RNN model accuracy on DPA Contest Dataset 54

4.13 RNN model loss on DPA Contest Dataset 55

4.14 CNN model loss on DPA Contest Dataset 56

4.15 RNN and CNN model performance comparison 56

4.16 Structure of the RNN model for Classification 57

4.17 Accuracy of the RNN model for Classification 58

4.18 Loss of the RNN model for Classification 59

4.19 Accuracy of the CNN model for Classification 60

4.20 Loss of the CNN model for Classification 61

4.21 Performance comparison of CNN and RNN 61

4.22 Structure of the RNN model for mapping keys without side-band 62

4.23 Structure of the RNN model for mapping keys with side-band 63

4.24 Accuracy of the RNN model for mapping keys without side-band 64

4.25 Loss of the RNN model for mapping keys without side-band 65

4.26 Accuracy of the RNN model for mapping keys with side-band 66

4.27 Loss of the RNN model for Mapping Keys with Side-band 67

4.28 Comparison of two RNN models 67

B.1 Experimental Setup for Collecting Physical Dataset 77

LIST OF ABBREVIATIONS

DES – Data Encryption Standard

AES – Advanced Encryption Standard

DSP – Digital Signal Processing

CNN – Convolution Neural Network

RNN – Recurrent Neural Network

CPS – Cyber Physical Systems

CFT – Continuous Fourier Transformation

CWT – Continuous Wavelet Transformation

DAQ – Data Acquisition

DCS – Distributed Control Systems

DFT – Discrete Fourier Transformation

DoS – Denial of Service

FFT – Fast Fourier Transformation

SVM – Support Vector Machine

CHAPTER 1

INTRODUCTION

Security in our lives is important, for our private information too. Cyberattacks, data breaching, illegal intrusion etc. are making our life hazardous and awful. More and more research should be done to cope with changes of exchanging information and to protect us from outsiders.

In this era, industries, businesses, and the general population generate, share, and exchange information as a form of data. These data values may reside in a computer, or a server or the cloud. Communication between two computers or two persons needs a secure channel, as people want their information to be secured and private. Securing communication, with different cryptographic implementations being used all over the world, is a matter of concern that information is being leaked by cyber-attackers. The main goal of these attackers is to break any security and to recover sensitive and private information that is supposed to be hidden, such as the passwords of individuals, credit card information, and device keys used for encryption or decryption. The problem does not rely on cryptographic algorithms. Instead, it relies on the cryptographic implementations of the devices or the systems. Side-channel attacks are the most popular kinds that attackers conduct to recover secret information from a device or a system without gaining full access to it. As the name suggests, side-channel attacks exploit side-channel leakage in the form of noise, the power consumption of a device, the behaviors of the registers, and so on. After getting any of these side-channel leakage information, the attackers search for methods to conduct the attack into the implementations of the device.

Side-channel attacks are alarming for this world because every walk of our lives depends on the usage of technology and sharing information to it. Nowadays, our information is preserved in the cloud, and we have no idea about the security of this information. We all want our information to be private and safe. Manufacturers of electronic devices come forward to make their products secure from side-channel attacks. Security experts and cryptologists are trying to come up with a solution to prevent these types of attacks on cryptographic implementations or devices to protect an individuals private information from leaking or being abused. This field of research is not only of academic interest but also manufacturers of cryptographic devices demand it, because there have been many examples of attacks on real-world cryptographic devices such as the bit-stream encryption in Xilinx FPGAs [1], the KeeLoq remote entry system [2], the YubiKey multi-factor authentication token [3], Mifare DESFire contactless payment cards [4], etc. These attacks made the manufacturers of these cryptographic devices and security experts aware of them.

We find very much interest in this area of research because there are lots of things to be done in this field of study. In our work, we have focused on differential power analysis attacks, which are the most popular form of side-channel attacks. In this work, we have used deep learning techniques to conduct our research on power consumption datasets, which are collected from cryptographic devices. As we know, deep learning is an advanced technique which is so efficient at learning by example. We have selected two popular models of deep learning neural networks. One is the Convolutional Neural Network (CNN), and the other one is the Recurrent Neural

Network (RNN). The short description of CNN and RNN models are given in the terminologies chapter.

The reason behind choosing these two models of deep learning techniques are:

- The characteristics of the datasets e.g., time series
- The volume of the datasets. RNN and CNN networks can handle large volumes of datasets efficiently.
- RNN and CNN, both work well with sequential data having various lengths.
- We want some predictive result from our datasets.

1.1 Thesis Statement

The main goal of this thesis work is to study side-channel attacks, especially differential power analysis attacks in a device which is using cryptographic algorithms implementation. Using deep learning techniques, we are interested in evaluating the performance of our neural network models in predicting the power traces (power consumption values) patterns to identify the known cryptographic algorithms such as Data Encryption Standard (DES). We know that DES algorithms have 16 rounds, so we want our models to predict the DES rounds and to differentiate datasets from non-DES. Moreover, after predicting the DES rounds and also classifying into DES and non-DES, we design our neural network models to guess on the keys. We have used the RNN model to recover the keys.

To research side-channel attacks, we answer the following important research questions (RQs):

RQ1: As an attacker, analyzing the datasets and feeding into the neural network, is it possible to detect which encryption algorithm the device is using?

RQ2: Is it possible to get the keys from the neural network model for a given plaintext-ciphertext pair?

RQ3: Using similar hyper-parameters, can we evaluate the performance of two neural network models (CNN vs. RNN)?

1.2 Our Major Contributions

In answering the above research questions, we have built two neural network models (CNN and RNN) to detect the underlying cryptographic algorithm of the device under attack. We have shown the comparison of the performance of these two networks. Our goal is to detect the underlying cryptographic algorithm that the target device is using and predict the keys for the encryption/decryption.

In our thesis work, we have several contributions.

- Comparison between CNN and RNN models on DPA contest Dataset: We have compared our two neural network models to predict the DES rounds from the power traces. Our datasets are DPA contest datasets in both cases. We are very interested in comparing two models using various performance metrics such as accuracy, speed, memory occupied, and loss of the models.

- Comparison between CNN and RNN models on Toggle Count Quantity Dataset: We have compared our models on toggle count quantity dataset. Using these toggle count values, we are interested to see how well the model performs. We also consider the same metrics that we use for DPA contest dataset.

- Comparison between CNN and RNN models in classifying the DES and non-DES algorithms: We have developed two network models (RNN and CNN) for binary classification of two classes: DES and non-DES. We want to evaluate the performance of each of the network models to see how well they can classify the two classes of cryptographic algorithms.

- Comparison between key mapping with and without toggle count quantity values: Using the RNN model, we compare the performance of key mapping with toggle count values and without toggle count values. For the first RNN model, our inputs are 64bit plaintext, 64 bits ciphertext, and one bit for encryption or decryption. We expect the output to be mapped with a specific key for a specific plaintext-ciphertext pair. For the other one, our inputs to the model are 64 bits plaintext, 64 bits ciphertext, one bit for encryption or decryption, and toggle count values. In each case, we are mapping our input to 56 bit key values. We evaluate the performance of each of the models using metrics such as the accuracy of the model, loss of the model, speed of execution.

For key detection for a given plaintext-ciphertext pair value, we have showed how accurately we can detect the key values.

1.3 Outline

The paper is organized as follows:

Chapter 2 discusses the background of the thesis work. In this chapter, we briefly describe the basic preliminaries or terminologies of Side-Channel Attack types, Data Encryption Standard (DES) algorithms, Basics of Digital Signal Processing (DSP), Deep learning techniques etc.

Chapter 3 discusses the related work in this field, and how our work is different from other work.

Chapter 4 broadly describes our methodologies of the thesis work.

Chapter 5 summarizes this thesis work with a conclusion section and possible direction for future studies.

CHAPTER 2

TERMINOLOGIES

2.1 Side-Channel Attack

Side-channel attacks can be defined as an attack where an attacker can gain sensitive information from the physical implementation of a cryptosystem. So, we can say that side-channel analysis is a branch of cryptography where information can be gained from the implementation of a electronic system, rather than the flaws in the cryptographic algorithms used. In this attack, attackers can gain information by exploiting the various sources such as timing information, power consumption, sound of the device, and electromagnetic leaks. There are many classes of side-channel attacks like cache attacks, electromagnetic attacks, timing attacks, software implementation attacks etc. Figure 2.1 shows the basic diagram of a side-channel attack in a cryptosystem. The attacker has generally no idea about the cryptographic algorithms used in the system or device, but getting ideas about the physical implementations, the attacker gains valuable information about the target.

In our thesis work, we work on power analysis attacks, which is one of the very special classes of side channel attacks. We discuss the power analysis attack in the next section.

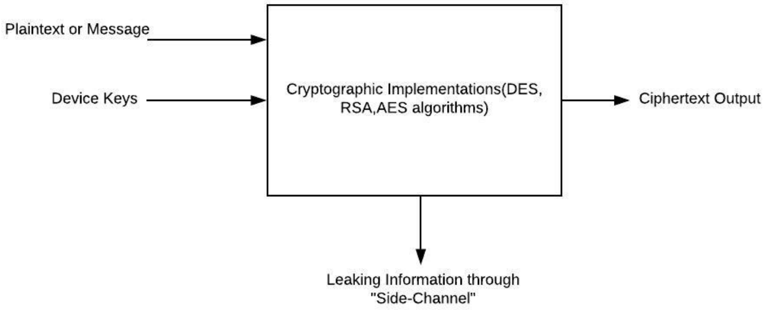

Plaintext or Message
Device Keys
Cryptographic Implementations(DES, RSA,AES algorithms)
Ciphertext Output
Leaking Information through "Side-Channel"

Figure 2.1: A typical diagram of Side Channel Attack

2.2 Power Analysis Attack

Power analysis attack is a class of side-channel attacks where an attacker uses the power consumption of the device or the system as the leaked information to exploit the target device or the system. The idea is that when the execution of a cryptographic algorithm (DES, AES, RSA etc.) on an electronic device occurs, then the device will consume power with a specific signature. The computation of the cryptographic algorithms takes a considerable amount of power. The attacker measures the power consumption of the device during the computation execution. These power consummations are then stored in a computer as power traces to do deeper analysis. The attacker tries all possible ways to retrieve information about the secret key in the algorithm from the captured power traces. After careful analysis using various methodologies, the attacker becomes successful in identifying all the sensitive information about the cryptosystem, as well as the devices key.

From Figure 2.2, we see the setup of a typical power analysis attack. Power analysis attacks usually consist of a target device under attack, a digital oscilloscope and a personal computer. The attacker hooks up an oscilloscope to the target device using probes. This target device is essentially executing one of the encryption schemes and running some kind of cryptosystem. The attacker measures and captures the power consumption of the target device at a determined sample rate. They get the collection of points which contains the voltage level measurements. Each of these measurements is called a power trace. These collection of power traces are stored into a personal computer. Using popular data processing applications (Python, MATLAB,etc), the attacker can analyze and process the collected power traces. The good thing is that the attacker does not to have any knowledge about the cryptographic algorithm being used by the target device. He/she only needs to make educated guesses and properly analyze the power traces to learn about the cryptosystem perfectly so that he/she can be able to conduct an attack.

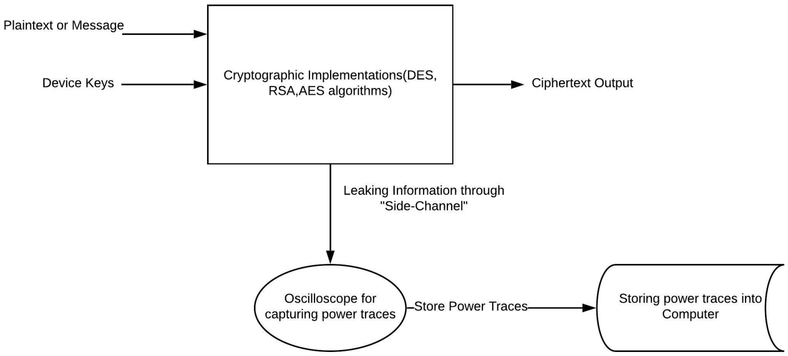

Plaintext or Message
Device Keys
Cryptographic Implementations(DES, RSA,AES algorithms)
Ciphertext Output
Leaking Information through "Side-Channel"
Oscilloscope for capturing power traces
Store Power Traces
Storing power traces into Computer

Figure 2.2: A depiction of Power Analysis Attack

Power analysis attacks are of different types such as simple power analysis (SPA), template-based analysis, differential power analysis (DPA) and horizontal power analysis (HPA). We discuss briefly to introduce two important types to the reader.

2.2.1 Simple Power Analysis (SPA)

SPA is the basic form of side-channel power analysis attack. In this attack, the attacker will observe and analyse one or more power traces and will try to get all possible information and reasoning of these traces. The attacker also tries to get information about the target system or the target device. Then, using the gathered information, he/she can be able to determine all the operations which are executed by the target device. Finally, he/she can be able to determine the cryptographic key.

Figure 2.3 shows the single power trace of an encryption algorithm. We can see the pattern of the dataset.

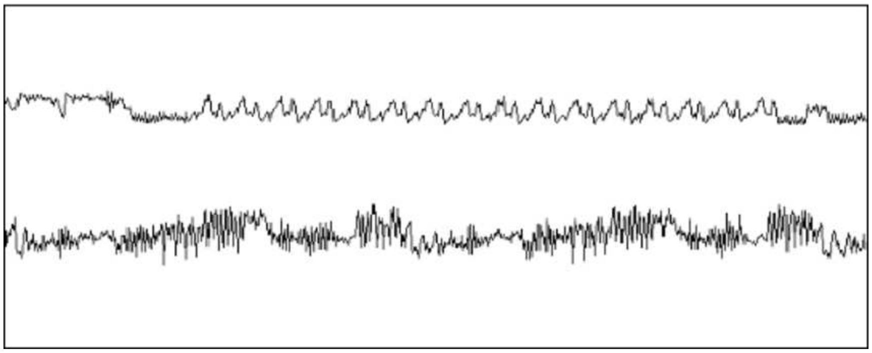


Figure 2.3: A single power trace of DES algorithm [19]

SPA is the technique to determine the knowledge about the target system and the secret key directly by analyzing one or a few power traces.

2.2.2 Differential Power Analysis Attack (DPA)

A DPA attack is also known as a non-profiled attack. Non-profiled attacks occur in closed devices such as smart cards of financial institutions, where an attacker does not have full control of the device. Hence, he/she can have a limited number of side-channel power traces of a cryptographic operation with a fixed unknown secret key value. In this attack, the attacker can have the following information from the device under attack:

1. A fixed secret key k where k belongs to a key space.

2. Random inputs/ Message

3. Random outputs/ Cipher-text

The attacker collects the side-channel traces and then combines all of this information to do mathematical analysis or key hypotheses using various well-known data analysis tools or algorithms to infer information to get the key.

In our thesis work, we will be working with differential power analysis attacks. We have our two datasets: one is physical data and the other is simulation data.

Figure 2.4 shows the basic structure of a differential power analysis attack in a cryptographic device.

In our dataset, we have closed device information that means side-channel traces that have fixed unknown keys, known messages, and known ciphertext.We analyze the traces and using two very well-known neural network techniques (RNN and CNN), we predict the pattern of the information to guess about the algorithm. We map the message-ciphertext pair with a specific key value in the key space.

2.3 Deep Learning Techniques

Deep learning is a very popular technique for machine learning algorithms. In deep learning, a model can learn by example and can imitate a human brain to get efficiency in doing or achieving something. Recent examples of deep learning are driver-less cars [20], digital image processing, pattern recognition, face recognition etc.

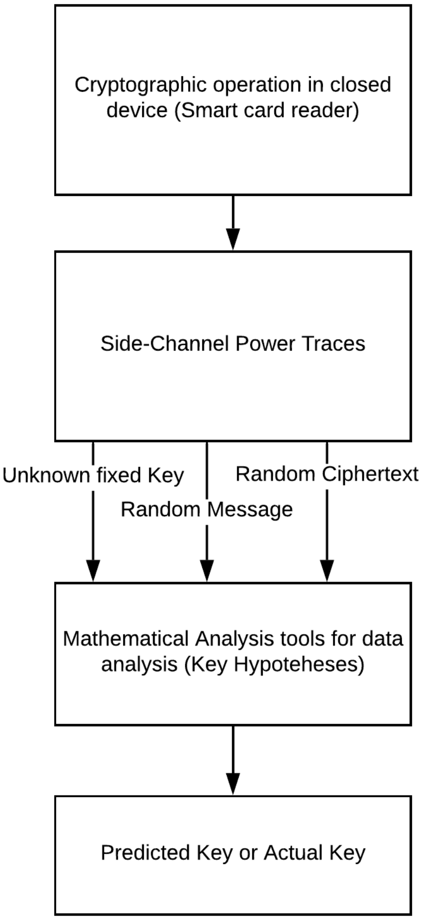

Cryptographic operation in closed device (Smart card reader)
Side-Channel Power Traces
Unknown fixed Key
Random Message
Random Ciphertext
Mathematical Analysis tools for data analysis (Key Hypoteheses)
Predicted Key or Actual Key

Figure 2.4: Differential Power Analysis Attack overview

In this section, we talk about deep learning neural networks. We have applied two neural networks (RNN and CNN) in our work.

2.3.1 Artificial Neural Network

An artificial neural network (ANN) is a network of artificial neurons. It is a structure where at least one input layer and one output layer should be presented. This input-output layer makes a neural network. Typically, there are other layers, named as hidden layers that may be present. When all the neurons of a neural network are connected to each other, we call it a fully connected neural network. There are basic terms to know to understand a neural network.

- Neuron: A neuron is a basic unit of the neural network. It takes input and multiplies the input by a specific weight and then adds bias values. After that, it passes the result through an activation function.

- Activation Function: Each neuron has an activation function. This function helps the neural network to achieve non-linearity. Having non-linearity, a model can create complex mappings of the data or can operate on complex input data such as videos, speech, and electrical signals. There are different types of activation functions. We have used four activation functions such as Sigmoid, TanH (Hyperbolic Tangent), ReLU (Rectified Linear Unit), and Softmax in our study.

- Epoch: An epoch is defined as a measure of the number of times we will pass our training input vectors through the learning algorithm or the model. In our work, we see that when we use fewer number of epochs, we achieve unexpected results due to under-fitting. So, we try to limit our epochs to 50-100 times. In each epoch, the weights are being updated and providing us better and better results. Input data values are passed forward and backward through the model in each epoch.

- Gradient Descent: Gradient descent algorithms are used to train the neural network by updating the weights [14]. In each iteration, using backpropagation, we calculate the derivative of the loss function with respect to each weight. Then, we subtract it from the weights. In this way, we update the weight values and train the model step by step.

- Learning Rate: Learning rate is a method to overcome the problem of overfitting in neural network model training. When we use gradient descent algorithms, our weights are updated, but the weights will change too far and will lead the model to overfit the training data. At the time of backpropagation, if we multiply learning rate with the derivative from of the gradient descent and then subtract it from the weights, we will get good results. We use learning rate value 0.02. In our work, keeping learning rate 0.02, we get good results when we train our model.

- Optimization Algorithm: These algorithms perform the gradient descent. While updating the weights and passing through the cost function or activation functions each time we calculate the error (actual data - predicted data). Our goal is to minimize the error. Optimization functions work to minimize the error by changing the parameters of the gradient descent of the neural network model. The most commonly used optimization algorithms are Adam, Adagrad, RMSprop, AdaMax, Nadam etc. In our work, we use Adam optimization algorithms, as we get more accurate results using it.

Figure 2.5 shows a structure of a typical neural network where there is one input layer and one output layer. There is no hidden layer.

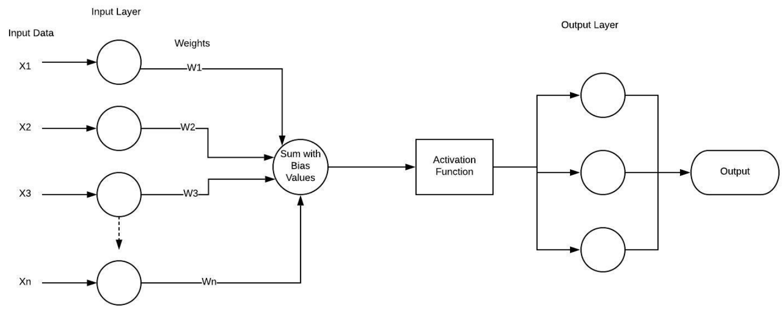


Figure 2.5: A simple Artificial Neural Network Structure

2.3.2 Convolution Neural Network (CNN)

Convolutional neural networks (CNN), or convNet in other terms, are very popular and widely used in image processing, analyzing images, data analysis and classifying different problems [13]. What makes CNN more unique than other neural network models is that it has convolutional layers instead of having typical hidden layers. These convolutional layers transform the input data to the model and pass it through the next convolutional layers. These transformation operations are called convolution operations. The assumption that a CNN model makes about the input is: the input are images and it encodes the properties from these inputs.

Each convolutional layer uses a series of filters or kernels to detect features in our intended images or patterns. These filters are n-dimensional arrays. The convolution operations make use of these kernels or filters. The filters are overlaid on the input data values and calculate the product of the filter and the input data to pass the output to the next neuron in the neural network. The output of the convolution operations are stored in an n-dimensioanl array called a feature map or an activation map. The more filters we apply, the more features we can detect and extract from the input data. This is the basic idea of how CNN works using convolutional layers to detect features from images.

When our data volume is large, such as when we have many images, then the operation of the convolution layers will be very time-consuming. In that case, we can apply pooling layers for convolution operation by reducing the number of parameters. Different pooling layers we can use are max pooling, average pooling and sum pooling.

Figure 2.6 demonstrates the operation of a CNN network. This figure is collected from paper [15].

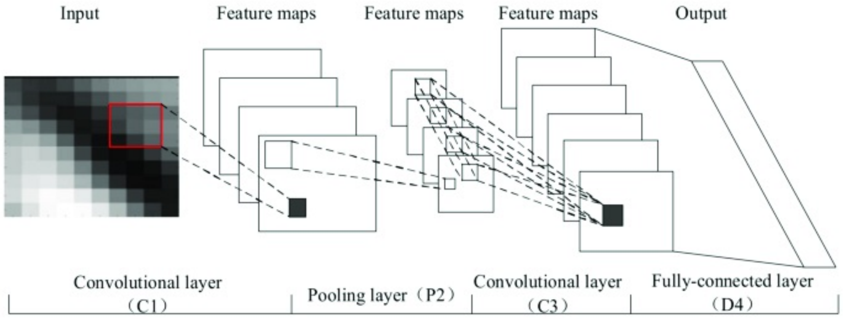


Figure 2.6: Overview of Convolutional Neural Network Operation

2.3.3 Recurrent Neural Network (RNN)

A recurrent neural network is a type of neural network model that was proposed in the 80s to model time series or to make use of sequential information. Why is this model so popular today? The answer is straightforward: for bigger data volume and bigger computation, RNN achieves a higher accuracy than machine learning techniques [12].

This network is called recurrent because, for every element of a sequence, it performs the same task where the output is dependent on the previous computations. The difference between the RNNs and CNNs is that any RNN-LSTM (long-short term memory) has a memory through which it can capture the information about what has been calculated so far.

A Recurrent neural network has a state and it basically receives input (input vectors) through time so that at every single time step we can feed an input vector into the network and it has some state internally and then it can modify that state as a function of what it receives at every single time step. There are synaptic weights inside the RNN which refer to the amplitude or strength of a connection between two nodes in a neural network layer. When we tune those weights, the RNN will have different behavior in terms of how its state evolves as it receives these inputs.

The formula for an RNN is:

$$h_t = f_w(h_{t-1} + x_t)$$

Here, Ht is the new state and Ht-1is the old state and Xt is the input vector as some time step t. Fw is denoted as a fixed function with weights (w) parameters.

The working procedure of an RNN model is generally the following:

- First, we provide input (input matrix) to the network (input could be anything).
- This input will be multiplied by the weight matrix and will also add a bias value.
- Then, this will pass through an activation function to activate the result of that layer and the result or output will feed into the next layer. The activation functions can be a sigmoid, tanH or reLu (rectifier linear unit). Actually, the output of the hidden layers is the result of a dot product operation and followed by a bias value. Bias values in a neural network are used to make a node in a layer always on. Bias values are set to 1 regardless of the data in a given pattern.

Backpropagation works by calculating gradients (multi-variable generalization of the derivative), which are needed in the calculation of the weights in a neural network. Every time, an RNN model goes through this process until it predicts the actual target output. If an error occurs in predicting, then we usually do back-propagation [10] to solve the error.

2.3.4 Long-Short Term Memory (LSTM) Network

These are the special types of neural networks which can solve the limitation of recurrent neural networks (RNNs). A typical RNN network faces difficulties in resolving the long-term dependencies in the input sequence. Using LSTMs, we can easily solve this problem because an LSTM network has capability to memorize the previous input for a long period of time. LSTM can learn long-term dependencies. This feature makes the LSTMs very useful when we want our RNN model to work with language modeling or text sequence processing.

A typical RNN has a chain-like, repeating structure. LSTMs also have this structure, but instead of having a single network layer, it utilizes four gate-like layers. These gates are very useful for avoiding vanishing gradient problems. These gates store information of backpropagation for future use and resolves long term dependencies.

2.3.5 Keras

Keras [16] is a library written in Python for high-level neural networks. It supports CPU and GPU acceleration libraries and enables a neural network model for fast prototyping. Using Keras, one can easily understand the structure of the model and build a model within a short period of time. Keras is compatible with Python version 3.6. We are using Keras version 2.2.4.

2.3.6 Tensorflow

Tensorflow [17] is a very useful framework for backends in a neural network. It is an open source platform for machine learning and deep learning network models. It enables us to easily build a model, to use complex machine learning functions, and to conduct powerful experiments for research.

2.3.7 Data Encryption Standard

Data Encryption Standard (DES) is a symmetric-key algorithm, which means it uses the same cryptographic key for encrypting a plaintext and decrypting a ciphertext. It is known as the block cipher and it operates on 64 bits messages. DES algorithms need a secret key of size 64 bits, where only 56 bits are used in encryption/ decryption and 8 bits reserved for parity. The DES algorithm for encryption or decryption works in the following way:

- First, 64 bits plaintext is run through an initial permutation (IP). After that, 64 bits messages break into two 32 bits halves.

- These two 32 bits halves will go through a Feistel network for 16 rounds.

- In the end, the final permutation will be done on the message, which is the inverse of the initial permutation of the plaintext.

- Finally, we will get ciphertext of 64 bits as the output.

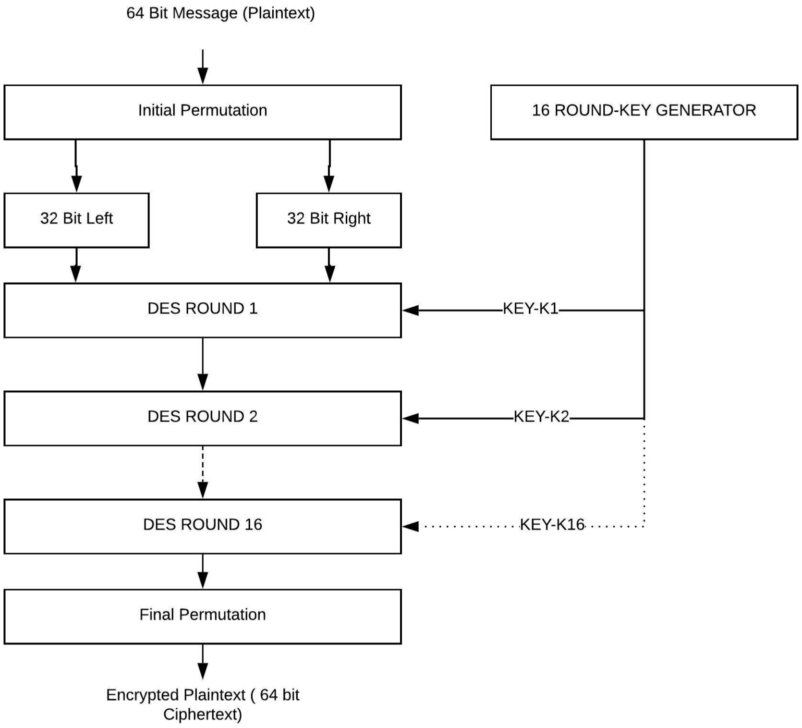


Figure 2.7: A pictorial view of DES algorithm

Figure 2.7 shows the big picture of the DES algorithm execution. We use DES algorithm in our work for side-channel data analysis, comparing the datasets with different neural network models.

2.3.8 Advanced Encryption Standard (AES)

AES is another type of symmetric encryption algorithm and much faster than triple DES. Like DES, AES also can be called a block cipher. It operates on 128 bits data and uses 128-256 bits keys.

AES works in an iterative manner not like a Feistel cipher. It actually accomplishes its computations on bytes rather than bits. Thats why plaintext messages of 128 bits block will be 16 bytes for AES and they are arranged in a 4x4 matrix format. The number of rounds in AES is not fixed. For example, 128 bits keys are needed for 10 rounds, 192 bits keys for 12 rounds and 256 bits keys for 14 rounds.

A typical encryption process is given in the Figure 2.8.

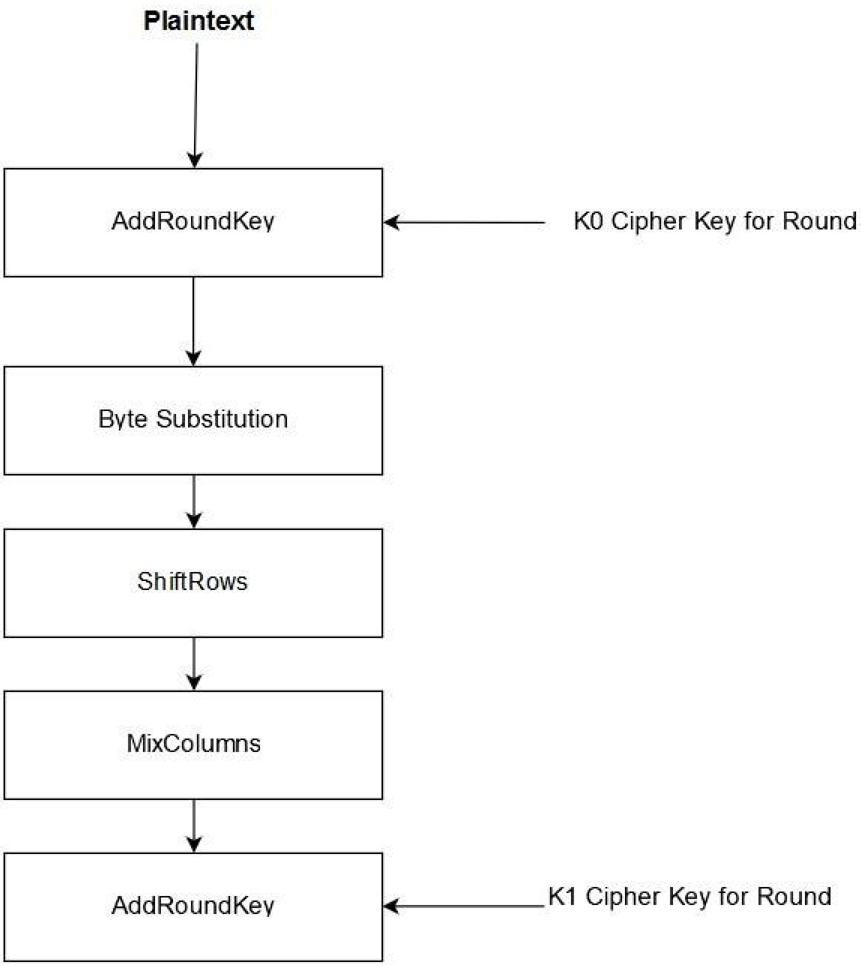


Figure 2.8: A pictorial view of AES algorithm

The decryption process of the AES algorithm is just the reverse operation of the encryption process. Nowadays, AES is used widely in both hardware and software. So, we use AES data values while doing binary classification of DES and non-DES.

2.4 The DPA Contest Dataset

Differential Power Analysis (DPA) contest [18] is a website where security researchers and students from different parts of the world participate to publish their attack modules in side-channel attack space. In this way, it is also a great source of publicly available datasets the people who are interested in working with differential power analysis attacks.

2.4.1 DPA Contest V1: DES Dataset

It is a contest organized and maintained by VLSI research group in France. This contest used data encryption standard (DES) to research DPA. We have collected the DES power traces from the DPA contest v1. We use the datasets in our thesis work to compare between CNN and RNN models.

CHAPTER 3

RELATED WORK

Recently, side-channel attacks have received much attention from researchers and security experts around the world. They use different methods of breaking ciphers to develop countermeasures, and after that they provide some ideas to strengthen them. Some of the researchers use deep learning models to perform side-channel attacks. Most of them used convolutional neural networks (CNNs) to demonstrate their attacks and to present efficiency of the attacks.

Samiotis et al. [6] covered a variety of classification scenarios to study CNN's performance on side-channel data. They examined the performance of CNN's on four different datasets of side-channel data and then compared their models with conventional machine learning (ML) algorithms and a CNN model from the literature. In our work, we have worked with CNN and RNN models. We have shown the difference between these two models in respect to performance and accuracy.

Maghrebi, Portigliatti, Prouff et al. [7] studied the application of deep learning techniques in the context of side-channel attacks for the first time. To evaluate their proposed attacks, they compared them to the most commonly used template attacks and machine learning attacks. They conducted their attacks on three different data-sets by evaluating the number of traces required. Finally, they showed their attacks outperform the state-of-art profiling side-channel attacks. We have studied side-channel attacks using two different datasets. In the future, we will enhance our research by conducting our research with more datasets.

Picek et al. [8] considered several scenarios for profiled Side-Channel Analysis (SCA) and compared the performance of several machine learning algorithms. They also suggested that if someone conducts profiled SCA, he should use convolutional neural networks (CNNs) for good results. However, we have shown that one can find an RNN model by changing hyper-parameters which will outperform the CNN model.

Kim, Picek, Heuser, Hanjalic, et al. [9] designed a CNN model with which they are able to achieve high performance with the random delay countermeasure. They suggested adding noise in the side-channel analysis evaluation while using deep learning techniques. We work with recurrent neural networks (RNNs) as we learned that for time series and sequential data, RNNs are the best fit [10]. So, we will evaluate our results to compare with other work that was done using CNNs or machine learning techniques.

Moreover, some of the researchers have applied machine learning (ML) techniques to leak information from unprotected and protected cryptographic implementations. Most of them focus on two popular methods, such as Support Vector Machine (SVM) and Random Forest (RF). Hospodar, Gierlichs, De Mulder, Verbauwhede et al. [11] studied the application of machine learning in the side-channel analysis. They focused on power analysis where power traces leak meaningful amounts of information about the processed cryptographic key. They also used LS-SVM as their classification technique. We have used two popular neural networks (CNN and RNN) to study power-analysis attacks.

CHAPTER 4

METHODOLOGY

We have worked with CNN and RNN models and applied these to two different datasets to compare the performance of the two models. We have evaluated our RNN model performance in predicting the DES rounds in DES algorithms. After predicting and getting an idea about the cryptographic algorithm, we have applied deep learning techniques (neural networks) to get information about the keys. We divide our work into three deep learning experiments and describe each of the experiments in this chapter.

4.1 Big Picture of the Thesis Work

We have done four comparisons: RNN vs. CNN on DPA contest datasets, RNN vs. CNN on toggle count quantity datasets, RNN vs. CNN on binary classification and RNN vs RNN on mapping keys. These comparisons are described as three deep learning experiments here.

Figure 4.1 shows the big picture of our work. We describe shortly each of these tasks in the later sections.

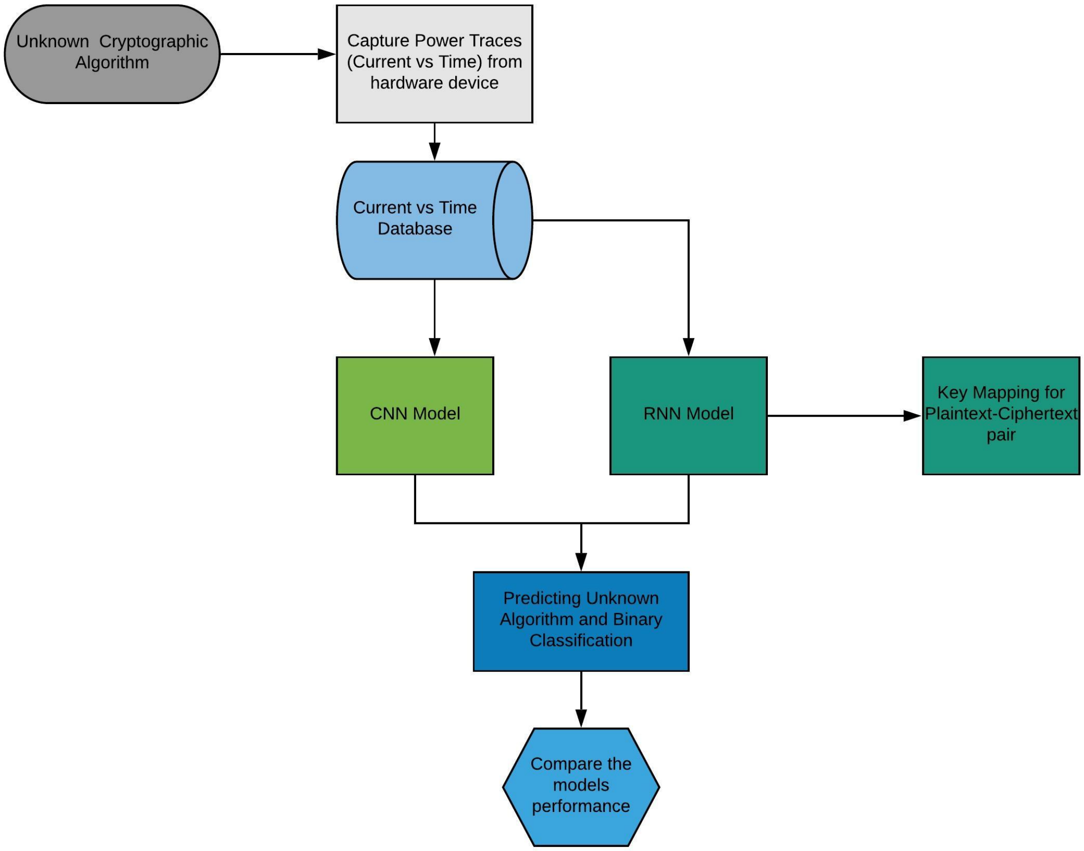

Unknown Cryptographic Algorithm
Capture Power Traces (Current vs Time) from hardware device
Current vs Time Database
CNN Model
RNN Model
Key Mapping for Plaintext-Ciphertext pair
Predicting Unknown Algorithm and Binary Classification
Compare the models performance

Figure 4.1: Overview of the hierarchical pattern of the thesis work

4.2 Datasets

We have worked with two datasets: publicly available datasets from DPA contest v1 and Simulated Toggle Count datasets from Verilog. We will be working on Physical Datasets from an FPGA board in our future work.

4.2.1 Dataset Structure

First we will talk about our datasets pattern, how we preprocess them, and how we prepare them for our model.

Table 4.1: Power Traces values are shown in the table

Power Traces	0.015917	0.015416	0.01739	0.01903	0.019447
Time	9.55E-07	9.55E-07	9.55E-07	9.55E-07	9.55E-07

The above table shows a few values of the power traces out of 50,000 traces. Here, we are only interested in the power traces values, and we will show how we preprocessed these values for the model to work.

The characteristics of data-sets are:

- The data-values are time-series data.
- Power trace values are the power consumed by the operation.
- After extracting the files from the DPA Contest v1 data-sets, we get thousands of CSV files containing power traces for a particular message, key, and ciphertext pairs.
- We have extracted the CSV files by running a C-program code that was given on the DPA contest website.
- Each CSV file has two columns: one is for time and the other is for power consumption value. The time here is not significant, as we are only interested in power traces, not in which time it was sampled.

The datasets we have used for our work are all time-series data. That means that all our series data points are sampled successively at equally spaced points in time. The sequences of the data points are related to the time it was sampled.

1. DPA Contest v1 Dataset: These are contained power traces values which are measured by an acquisition platform. These traces are collected and stored

by and belongs to Telecom ParisTech. The power consumption traces data stored in '.bin' format. They have also provided a C-program file named as 'agilent_bin_reader.c' to convert '.bin' file into a '.csv' file.

DPA Contest V1 Dataset		
Filename	Total Traces Count	File Size
secmatv1_2006_04_0809.zip	81,089	4 Gbytes
secmatv3_20070924_des.zip	81,569	1 Gbytes
secmatv3_20071219_des.zip	67,753	6 Gbytes

Table 4.2: Information of DPA Contest Dataset

Power consumption traces are presented in the datasets as voltage. The voltages are in nanovolts (nV $1e-9$) units.

2. Simulated Dataset: We worked with simulated datasets from Verilog. These datasets files contained time and toggle count. Dataset information is based on specific plain-text, cipher-text pairs and also their corresponding keys.

Toggle Count Quantity Dataset		
Filename	Total Traces Count	File Size
0_key_message_ciphertext.csv	49	5 KB
1_key_message_ciphertext.csv	49	5 KB
0_key_message_ciphertext.csv	49	5 KB
1_key_message_ciphertext.csv	49	5 KB
0_key_message_ciphertext.csv	49	5 KB

Table 4.3: Toggle Count Datasets from Verilog

In these datasets, filename is composed of encryption or decryption (0 or 1), Plain-text, Cipher-text, Keys. The contents of the files are time and toggle count data. In Table 4.1, the structure of the datasets is shown.

3. Experimental Dataset: These datasets are collected from the Artix-7 FPGA board. The evaluation kit of Artix-7 has single and double differential I/O standards.

4.2.2 Dataset Cleaning (Munging) and Visualization

In this section, we will talk about the data cleaning process of our three datasets. Our datasets contain noisy data values, irrelevant data points, etc. So, our target is to clean our datasets before we will proceed training and validating these in neural networks.

For cleaning the datasets, we have applied digital signal processing concepts. The methods we used are Hamming, Hanning window functions.

The following figures show the effect of using digital signal processing methods in our side-channel DPA contest datasets.

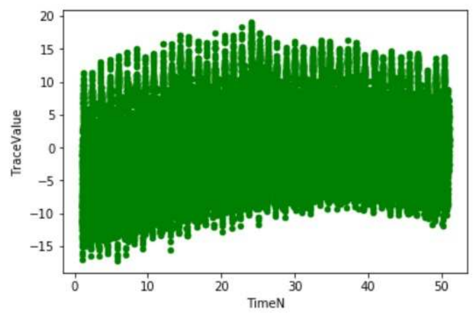


Figure 4.2: Initial condition

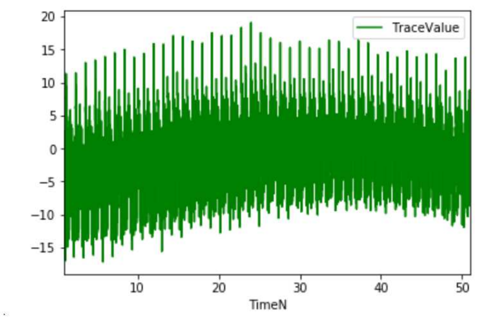


Figure 4.3: After Cleaning

After applying Hamming, Hanning window functions we get Figure 4.4 of DPA contest datasets. In the figure, the x-axis represents the power traces in voltage and the y-axis represents the time. We see that most of the noise has reduced and the leakage of the datasets are visible.

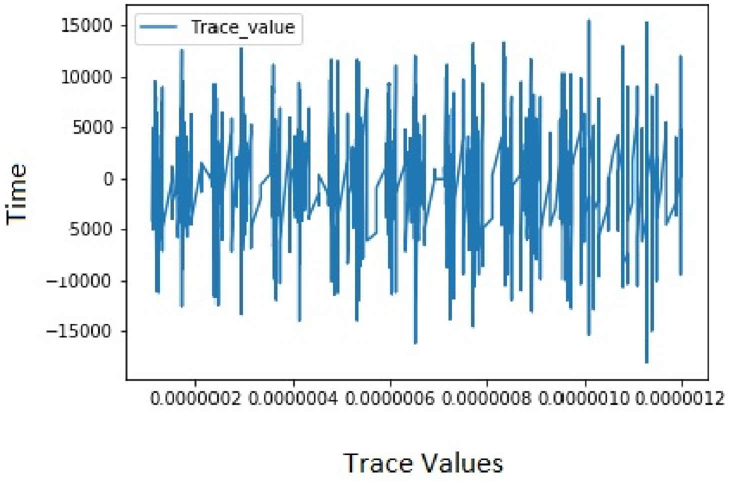


Figure 4.4: DPA contest dataset after applying windowing functions

We need not apply the digital signal processing methods to the toggle count datasets because they are simulated data values from Verilog HDL. The following Figure 4.5 shows the toggle count datasets patterns. From the figure, we see that when there is a change found in the data, it shows a high spike.

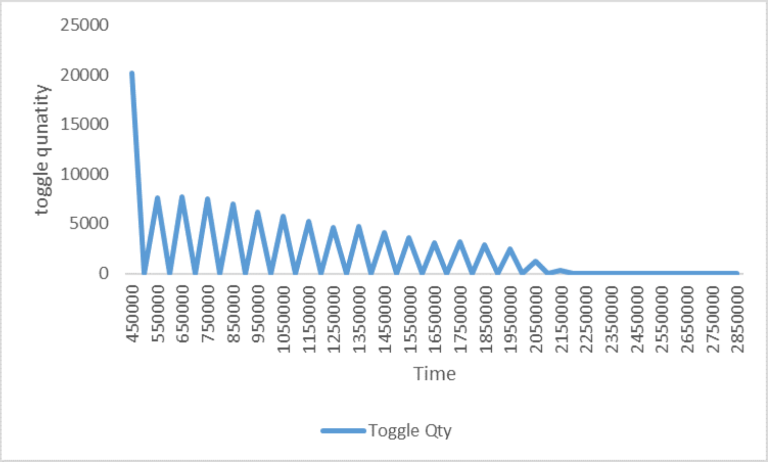


Figure 4.5: The toggle count dataset pattern

4.3 Deep Learning Experiments

We have done three different deep learning experiments with DPA contest and toggle count datasets. We have used CNN and RNN as our deep learning techniques.

1. Experiment 1: In this experiment, we show the comparison of RNN and CNN models in prediction DES rounds from DPA contest and toggle count datasets. We measure the loss and accuracy of the two models. We evaluate the speed of the execution of these two models while processing the same amount of datasets.

2. Experiment 2: Here, we classify our datasets into two classes: DES and non-DES. Doing so, we use RNN and CNN models. We evaluate the performance of these two models and perform the comparison.

3. Experiment 3: In the last experiment, we evaluate the performance of two RNN models in mapping the specific key for a plaintext-ciphertext pair based on with or without side-band information.

4.4 Experiment 1: Prediction of DES rounds

We know that the DES algorithm operates with 16 rounds for encryption and decryption. In this section, we discuss the prediction of DES rounds using deep learning techniques. We have predicted the DES rounds with both CNN and RNN models. We predict DES rounds on DPA Contest and toggle count datasets. Finally, we compare the model and model accuracy between CNN and RNN models. Again, we are repeating that our main goal is not to get higher accuracy using an RNN model rather than a CNN model. We are interested in how well our RNN model works, and we try to get the accuracy closer to the CNN model.

Here we consider that the attacker has only power traces. With only the power traces of a cryptographic device, how can he/she be able to predict the DES rounds and gain some knowledge about the cryptographic algorithms?

4.4.1 RNN Model to Predict on DPA Contest Data

We have developed a recurrent neural network model to predict the Data Encryption Standard (DES) rounds. First, we consider 1000 samples of the data to see how it can predict the subsequent rounds and what it looks like.

Figure 4.6 shows the DES rounds prediction on DPA contest datasets with only 1000 traces. The x-axis represents the number of traces and the y-axis represents the values of the trace.

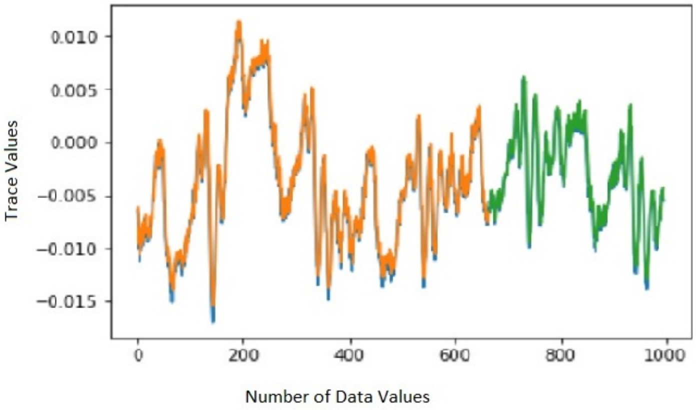


Figure 4.6: DES rounds prediction with 1000 traces

Then we predict the DES rounds with 5000 traces. Here, we split the dataset 67% for training and 33% for validation and test purposes. We found that the attacker can get an idea about the algorithm through a thorough analysis of the signals.

Figure 4.8 shows the DES rounds prediction on DPA contest datasets with only 5000 traces. The x-axis represents the number of traces and y-axis represents the values of the trace.

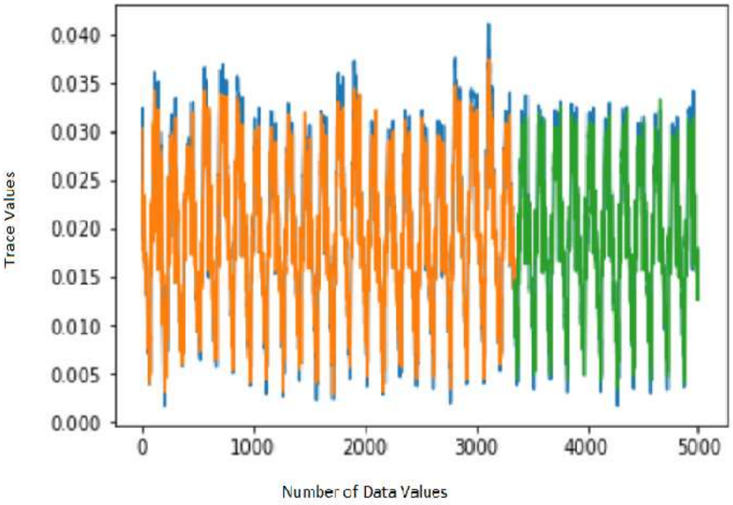


Figure 4.7: DES rounds prediction with 5000 traces

Finally, we operate our model on 50,000 trace values to have some idea about the patterns of the datasets and also predict the DES rounds. In doing so, we split our datasets in our RNN model, 67% for training and 33% for testing. We will show our model accuracy and loss in the subsequent subsections.

Figure 4.8 shows the DES rounds prediction on DPA contest datasets with only 5000 traces. The x-axis represents the number of traces and y-axis represents the values of the trace.

4.4.2 RNN model on Toggle Count Datasets

We have applied our RNN model to predict the subsequent rounds in the toggle count datasets based on the toggle quantity and time. As these datasets are time series, we

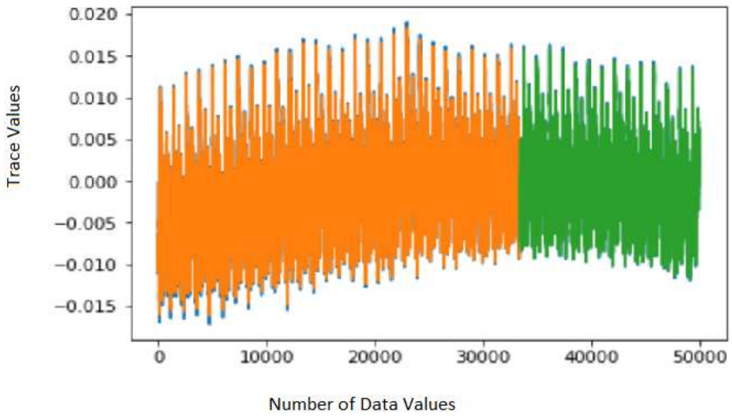


Figure 4.8: DES rounds prediction with 50000 traces

can easily apply the RNN model on these datasets to predict future values or more patterns.

Figure 4.9 shows the DES rounds prediction on DPA contest datasets with only 5000 traces. The x-axis represents the number of traces and the y-axis represents the values of the trace.

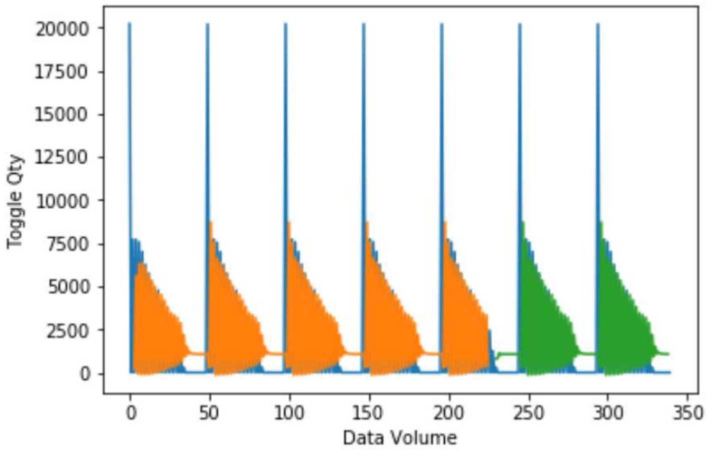


Figure 4.9: DES rounds prediction with 350 sets of Toggle Count traces

4.4.3 Baselines

We compare the RNN and CNN models' performance using several metrics such as baseline, number of epochs, speed of execution, accuracy and loss of the model.

Here, we do not classify our datasets into different classes, so the random baseline will be 100% based on the definition. According to random baseline definition, all our 50,000 key, ciphertext and plaintext values are unique and random.

On the other hand, our most-common baseline will be 0% because we do not have common key, ciphertext and plaintext values.

4.4.4 Result: Accuracy and Loss Evaluation in Predicting

In this section, we discuss the evaluation of the RNN model on the prediction of DES rounds of both DPA contest dataset and toggle count datasets. We show a comparison of value loss between and RNN and CNN models. We make the same prediction with CNN models. We observe CNN and RNN model accuracy and loss in predicting the subsequent DES rounds.

RNN Model Accuracy on Toggle Quantity Count Datasets

In this subsection, we show the accuracy and loss that we achieved after running our RNN model on toggle quantity count datasets.

Figure 4.10 shows the DES rounds prediction accuracy of our RNN model on the toggle quantity datasets. The x-axis represents the number of epochs and the y-axis represents the accuracy in percentage.

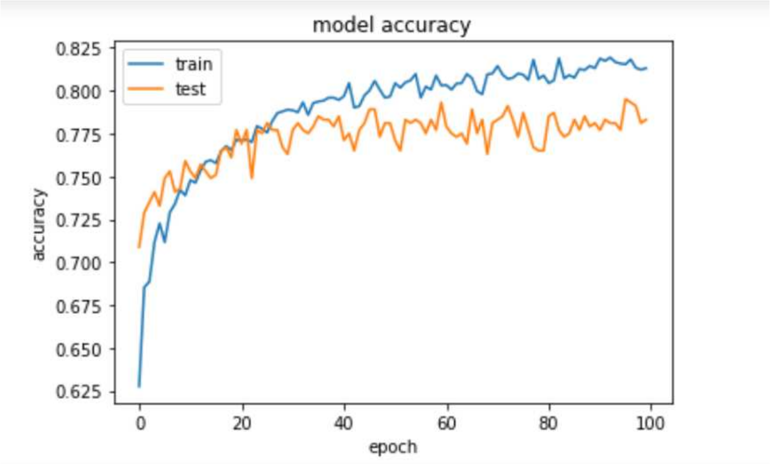


Figure 4.10: RNN Model Accuracy on Toggle Quantity Count Dataset

In Figure 4.10, we can easily see that while training the model it achieves almost 82.5% accuracy. For testing over the rest of the datasets, it achieves almost 80% accuracy. We have run our model up to 100 epochs.

RNN Model Loss on Toggle Quantity Count Datasets

Here, we show the loss of the model in predicting the DES rounds in the toggle quantity count datasets.

Figure 4.11 shows the DES rounds prediction loss of the RNN model on the toggle Quantity datasets. The x-axis represents the number of epochs and y-axis represents the loss in percentage.

From Figure 4.11 we can easily see that while training and testing the datasets, the loss is decreasing depending on the number of epochs of the neural network model. At the time of training, the loss of the model falls sharply. While testing the datasets, the loss decreases frequently.

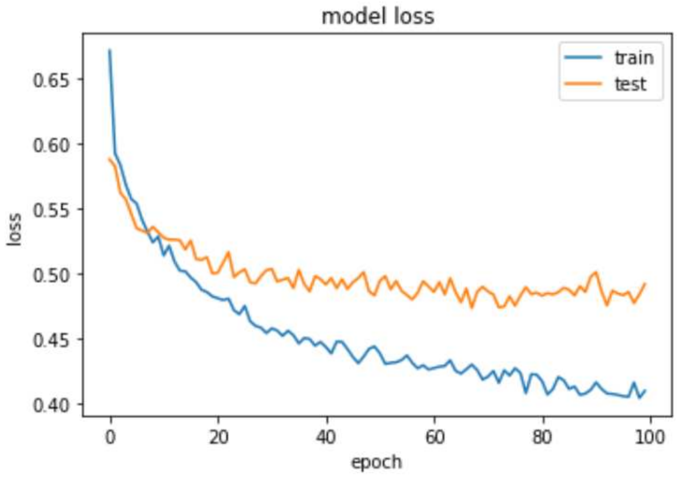


Figure 4.11: RNN Model Loss on Toggle Quantity Count Dataset

RNN Model Accuracy on DPA Contest Dataset

In this subsection, we show the accuracy and loss that we achieved after running our RNN model on DPA contest datasets.

Figure 4.12 shows the DES rounds prediction accuracy of our RNN model on the DPA contest dataset. The x-axis represents the number of epochs and y-axis represents the accuracy in percentage.

From Figure 4.12 we can easily see that while training the model it achieves almost 78% accuracy. For testing over the rest of the datasets, it achieves almost 76% accuracy. We have run our model up-to 30 epochs. We have run our model for DPA contest dataset up-to 30 epochs in order to make simple and save computation time.

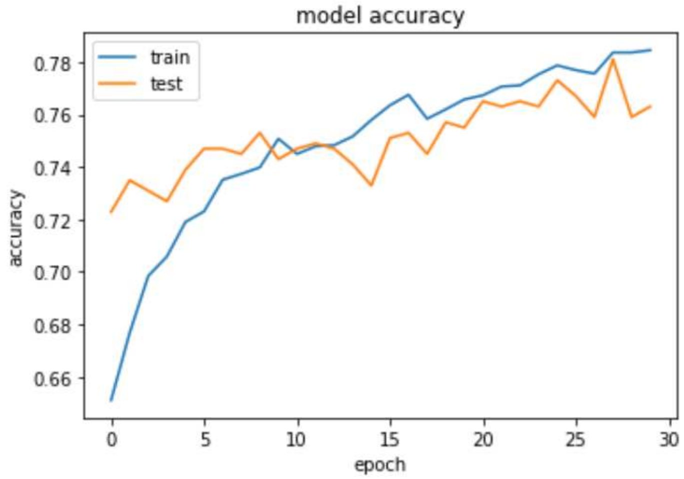


Figure 4.12: RNN model accuracy on DPA Contest Dataset

RNN model loss on DPA contest Dataset

Here, we show the loss of the model in predicting the DES rounds in the DPA contest datasets.

Figure 4.14 shows the DES rounds prediction loss of RNN model on the DPA contest datasets. The x-axis represents the number of epochs and y-axis represents the loss in percentage.

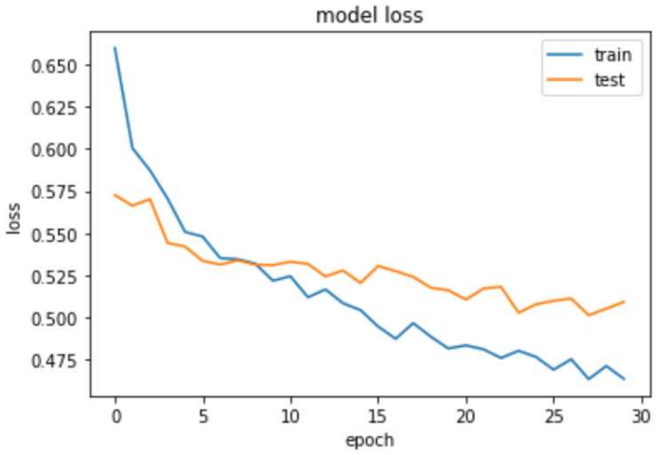


Figure 4.13: RNN model loss on DPA Contest Dataset

From Figure 4.14 we can easily see that while training and testing the datasets, the loss is decreasing depending on the number of epochs of the neural network model. After 30 epochs, we see the value loss reduces to less than 47.5% for each training and testing of the model.

Comparison of RNN and CNN model loss

Here, we discuss the efficiency of the RNN model and CNN model in predicting the DES rounds with both toggle count datasets and DPA contest datasets.

Figure 4.14 shows the DES rounds prediction loss of the CNN model on the toggle Quantity datasets. The x-axis represents the number of epochs and the y-axis represents the loss in percentage.

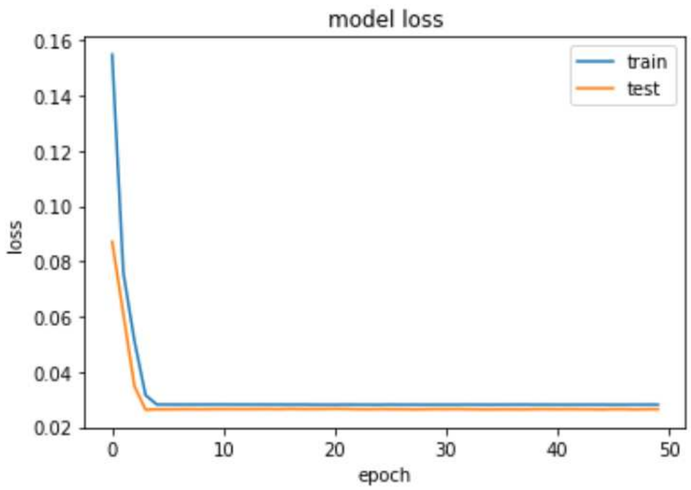


Figure 4.14: CNN model loss on DPA Contest Dataset

From Figure 4.14 we can easily see that while training and testing the datasets, the loss is decreasing depending on the number of epochs of the neural network model. After 6 epochs, we see the value loss remains constant up-to 50 epochs. The loss of the model is about 4%. In this case, it is better than our RNN model performance.

Here, we see better accuracy and less loss during training and testing of the CNN model, because CNNs can deal with misaligned traces, as proposed in [5].

While in our RNN model, we have used the LSTM network, whose next hidden layer output mainly depends on what it sees in previous layers.

4.4.5 Discussion

We run 50 epochs for our RNN and CNN models. For each epoch it takes 1 second for our CNN model and 2 seconds for our RNN model. So, we can say the CNN model's execution has better speed than the RNN model for predicting the DES rounds.

Figure 4.15 shows the comparison between the performance of our RNN and CNN models.From the figure, we see that the CNN model performs better than the RNN model in predicting DES rounds.

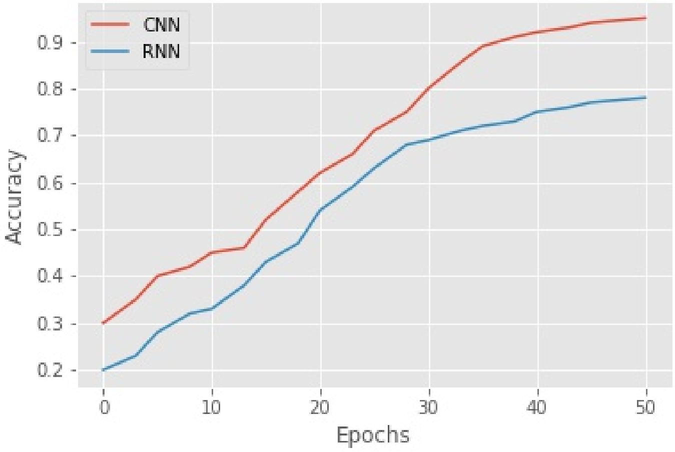


Figure 4.15: RNN and CNN model performance comparison

Table 4.4 represents the comparison of the CNN and RNN models in predicting DES rounds with toggle quantity count and DPA contest datasets.

Result Comparison I				
Model	Speed (Time per epoch)	Training Accuracy	Validation Accuracy	Loss
CNN	51 seconds	98%	95%	4%
RNN	100 seconds	82%	78%	38%

Table 4.4: Comparison of CNN & RNN on different matrices

4.5 Experiment 2: Classification of Two Classes: DES and Non-DES

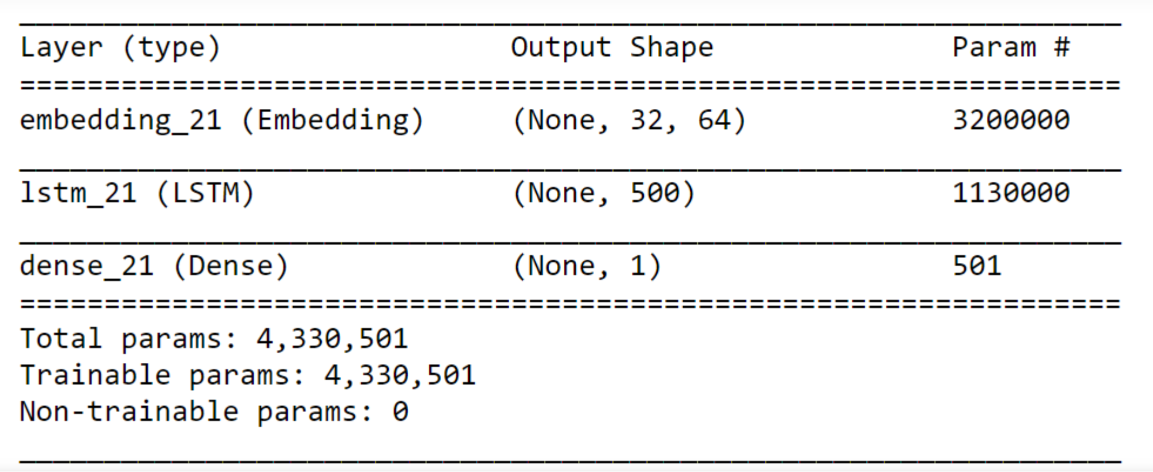
In this section, we discuss the classification of DES and non-DES classes using the side-channel datasets. Here, we assumed that the attacker is powerful enough to get the plaintext, ciphertext, and key values for specific plaintext-ciphertext pairs. Now, the attacker will get the traces from other cryptographic devices, and he is interested in classifying his/her datasets into DES and non-DES classes using deep learning techniques. We have chosen the RNN model and LSTM network to classify the datasets into two classes: DES and non-DES.

4.5.1 Procedure of the Experiment

Here, we have 68,000 DES datasets and 55,000 non-DES datasets (AES datasets). The main goal is to classify the datasets into two classes. For AES datasets we want our model to label it as 0, which means non-DES. And for DES datasets the label will be one (1), which means it is a DES dataset.

We have split our datasets into 60% for training and 40% for validation. Our plaintext, ciphertext, and keys are in text format. So, using the embedding layer, we have converted the text into vectors for computation into the neural network.

Figure 4.16 shows the structure of the recurrent neural network for completing the binary classification task.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_21 (Embedding)     (None, 32, 64)            3200000
_________________________________________________________________
lstm_21 (LSTM)               (None, 500)               1130000
_________________________________________________________________
dense_21 (Dense)             (None, 1)                 501
=================================================================
Total params: 4,330,501
Trainable params: 4,330,501
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.16: Structure of the RNN model for Classification

The task is a binary classification task. We have used binary_crossentrophy as our loss functions and Adam optimizer as optimization functions. Here, we are interested in the accuracy metrics. We have found almost 86.51% accuracy in classifying DES and non-DES classes using our RNN model for binary classification. We have labeled the datasets based on the feature that DES contains 16 characters hexadecimal values in plaintext, ciphertext, and keys. The idea is if the length of the plaintext or ciphertext is equal to 16, we labeled it as 1, which means DES, and 0 otherwise. Binary classification accuracy for classifying the two classes: DES and non-DES is good in this case.

4.5.2 Baselines

We consider two baselines: random and most common baselines of our datasets. The metrics that we consider while comparing the two models' performance are: speed of execution, loss and accuracy of the models for a fixed number of epochs etc.

- Random Baseline: The random baseline in this case is 50% because we are using two classes: DES and non-DES. For DES, the model will classify it as 1 and for Non-DES (AES), the model will classify it as 0.

- Most Common Baseline: For the classification task, we have 68,000 DES datasets and 55,000 non-DES (AES) datasets. The most common baselines is 55.28%. For most common baselines, we have divided 68,000 DES datasets with the total 123,000 datasets.

4.5.3 Result Comparison of RNN and CNN in Classifying Two Classes

We have done the binary classification using the CNN and RNN model to check the performance between the two models. We have measured the model accuracy and model loss for the two models and showed which model performs well for the binary classifying task. The comparison of these two models is shown in the following sections.

RNN Model Accuracy and Loss

We use the same hyper-parameters for two models to compare the performance. We use LSTM network, 'sigmoid' as activation function, 'rmsprop' as optimizer, and 'binary_crossentrophy' as loss function. We measure the accuracy for the model in both training and validation cases.

According to Figure 4.17, the accuracy of the model is 86.51%. That means it can classify the DES data values from the other (AES) data values with good accuracy.

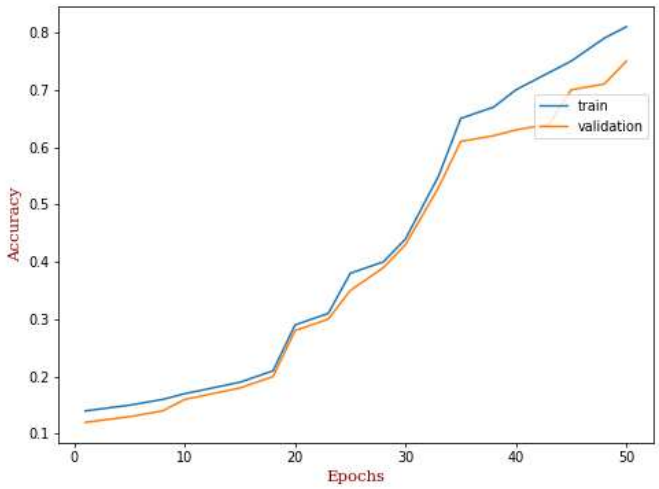


Figure 4.17: Accuracy of the RNN model for Classification

On the other hand, the model has loss as well. The loss of the model has a zigzag fall. It maintains a constant value at 18.23%. We can see the loss of the model in Figure 4.18 for both the training and validation cases.

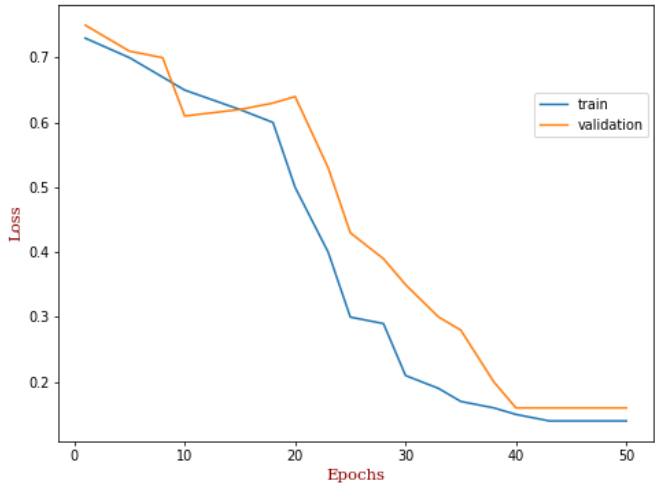


Figure 4.18: Loss of the RNN model for Classification

CNN Model Accuracy and Loss

We keep the hyper-parameters the same for performance evaluation. For the CNN model, we have better accuracy than the RNN model. The CNN model works good in classification tasks.

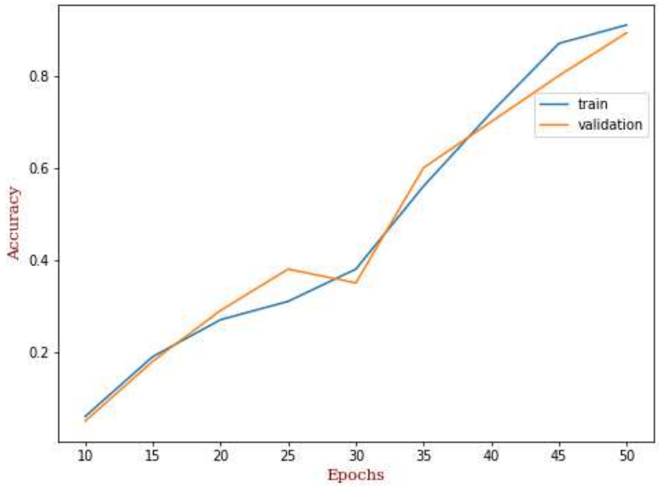


Figure 4.19: Accuracy of the CNN model for Classification

We have almost 90% accuracy for the CNN model in classifying the DES and Non-DES data values. Figure 4.19 shows the accuracy of the CNN model.

The loss of the CNN model is shown in Figure 4.20. The loss of the model tends toward 10%.

We have the loss for the models in both cases: training and validation. Training loss gives us the information about error occurred during training.

4.5.4 Discussion

Our CNN model outperforms the RNN model in classifying DES and non-DES classes. Figure 4.21 shows the accuracy curve of the two models which proves that CNN

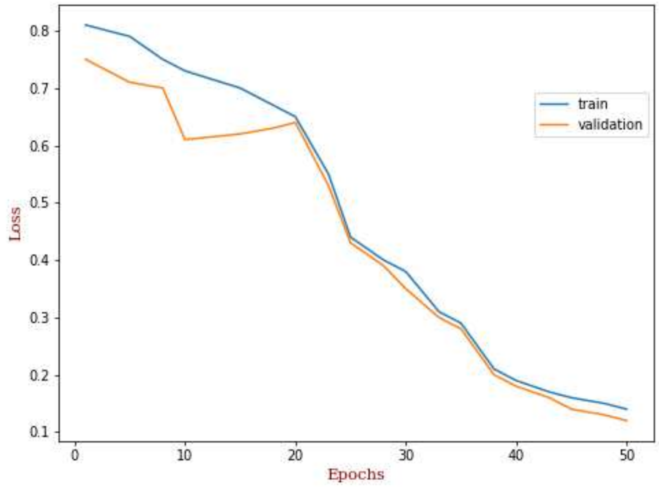


Figure 4.20: Loss of the CNN model for Classification

performs well. We present the information in tabular form also.

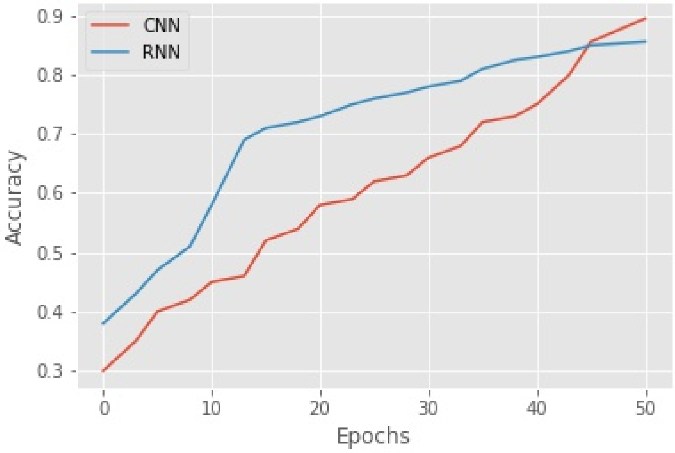


Figure 4.21: Performance comparison of CNN and RNN

The following table 4.5 represents the performance comparison between the CNN and RNN models at a glance.

From the table, we can get the idea of the accuracy and loss of the two models

Result Comparison II				
Model	Speed (Time per epoch)	Training Accuracy	Validation Accuracy	Loss
CNN	110 seconds	89.9%	89.5%	10%
RNN	132 seconds	87%	85.6%	38%

Table 4.5: Comparison of CNN & RNN on Binary Classification

and the speed of the execution. It takes time for both of the models because of large training and validation datasets. We get better accuracy for the CNN model in the binary classification. An attacker can utilize the neural network model and can guess the algorithm used in the system. This will lead the attacker to attack the device or system with less effort.

4.6 Experiment 3: Recurrent Neural Network Model for Mapping Keys

In this section, we use the RNN model to get the key values for a specific plaintext-ciphertext pair.

We show two different works: in the first one, we input plaintext, ciphertext and encryption/decryption bits (0/1) to the neural network. The output of the neural network model will map to the exact key for a specific plaintext-ciphertext pair.

For the next one we input plaintext, ciphertext, side-band information (toggle count quantity) and encryption/decryption bit (0/1) to the neural network and the output we expect keys mapped in 56 bits key-space.

After that, we evaluate the performance of key mapping with side-band and without side-band.

4.6.1 Procedure of the Experiment

Key Mapping without Side-band Information

We first build an RNN model which will take three inputs: plaintext, ciphertext and one bit encryption or decryption key. Plaintext and ciphertext are in 16 byte hexadecimal form. The working procedure of the RNN model can be summarized below:

- First, we convert the data from hexadecimal to binary values. So, we have 64 bits binary plaintext, 64 bits ciphertext and one bit encryption/decryption key. The input of the network is 129 bits binary values.

- Then we simply think about the multi-class logistic regression model of machine learning where if you input two or multiple elements, it will be mapped with the respective third element. So, we have used softmax in the final layer. As we know, softmax generates the probabilities for a specific input to the specific output.

- Here, we want our model to map with exact key values for a specific plaintext-ciphertext pair.

- Final model output is 56 bits key for a plaintext-ciphertext pair.

Figure 4.22 shows the logical flow of the task for key mapping without side-band. Here, side-band information means the toggle count quantity information that we get from Verilog HDL simulation. For the next key mapper RNN model, we use side-band information along with the other inputs.

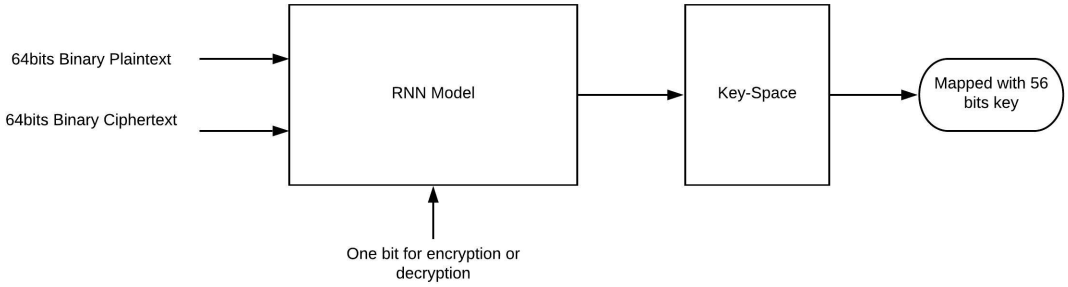


Figure 4.22: Structure of the RNN model for mapping keys without side-band

Key Mapping with Side-band information

In this key mapping task, we use the same methodology for getting a key value for a specific plaintext-ciphertext pair. The difference is in the input parameters. Here, our inputs to the model are plaintext, ciphertext, encryption/decryption key, and side-band information (toggle count quantity data values). The reason behind using side-band information as the extra input parameter in the RNN model is that we are interested to see if it is possible to achieve better accuracy or to get better result than without having side-band information in the input. The work procedure is as follows:

- Like the previous task, we have converted our plaintext, ciphertext from hexadecimal to binary values. With one bit encryption/ decryption key, we have 129 bits binary input to the neural network model. Additionally, we have side-band information along with these inputs.

- In the same manner, we use softmax before the output layer. Softmax gives us probabilities for multi-class regression. For a specific plaintext-ciphertext pair and having additional information, the model mapped with a key value from the key-space.

- The output of the model is 56 bits key for the specific inputs.

Figure 4.23 shows the structure of the inputs and output of the RNN model to map the key values with using side-band information.

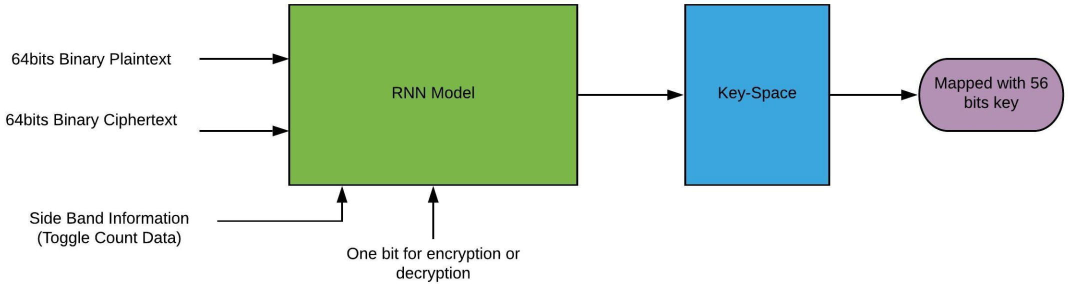


Figure 4.23: Structure of the RNN model for mapping keys with side-band

4.6.2 Baselines or Metrics

The metrics that we use in this experiment to evaluate the performance of two RNN models are: speed of the execution, loss and accuracy of the models for a specified number of epochs, and memory required to store the model etc.

Our datasets for this experiment are generated from python DES programs. In that programs, each time we generate random sets of plaintext-ciphertext pairs and also a specific key for the pairs.

- Random Baseline: The random baseline in this case is 100% because our ciphertext-plaintext pairs are randomly generated and also the key for these pairs are randomly generated.

- Most Common Baseline: For the classification task, we have 76,000 DES datasets. The most common baselines is 0%, because all the pairs of ciphertext-plaintext are random.

4.6.3 Performance Evaluation of RNN Model with and without Side-band Information

The key-mapping task is done by using recurrent neural network (RNN) only. We have not tested it with convolutional neural network (CNN). So, the performance of the RNN model will be evaluated for the two different cases. For the first case, we will evaluate the RNN model performance without side-band information. And for the second case, we will use side-band information for key-mapping.

Evaluation of Model Accuracy and Loss without Side-band

We have evaluated the RNN model accuracy for the training and validation datasets. From Figure 4.24, we can see that the accuracy of the RNN model is almost 79%.

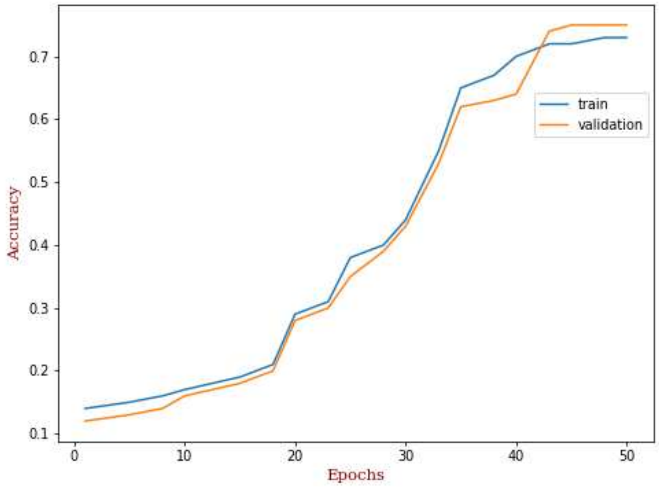


Figure 4.24: Accuracy of the RNN model for mapping keys without side-band

We have evaluated the loss of the model for the training and validation datasets. Figure 4.25 shows the loss of the model. We can see that the loss of the model maintains a constant loss value at 23% after a zigzag decrease.

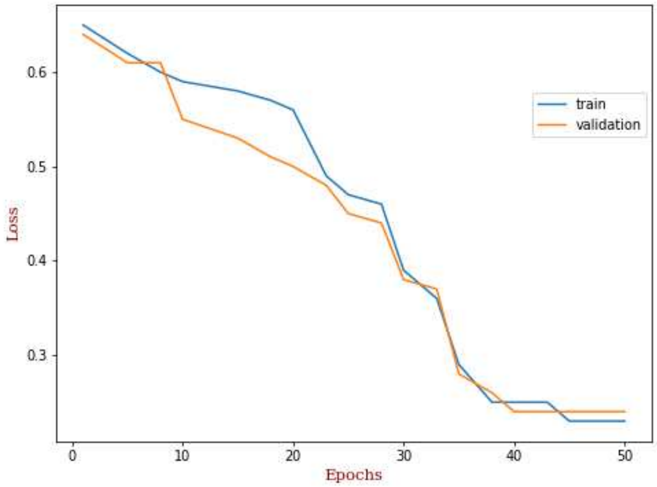


Figure 4.25: Loss of the RNN model for mapping keys without side-band

Evaluation of Model Accuracy and Loss with Side-Band

We expected the better accuracy of the RNN model while we have additional information such as side-band information. We have achieved better accuracy, as was our expectation.

Figure 4.26 shows the accuracy of the model after using the side-band information with our datasets.The accuracy of the model is almost 87.3% in this case. So, we got better accuracy with side-band information.

The loss of the model was less in this case than the loss without side-band information. From Figure 4.27, we can see that the loss of the model is 19%.

4.6.4 Discussion

We have run our RNN model for 50 epochs. For the first case, it takes 2.5 seconds per epoch. And for the second case, it takes 3.2 seconds per epoch.

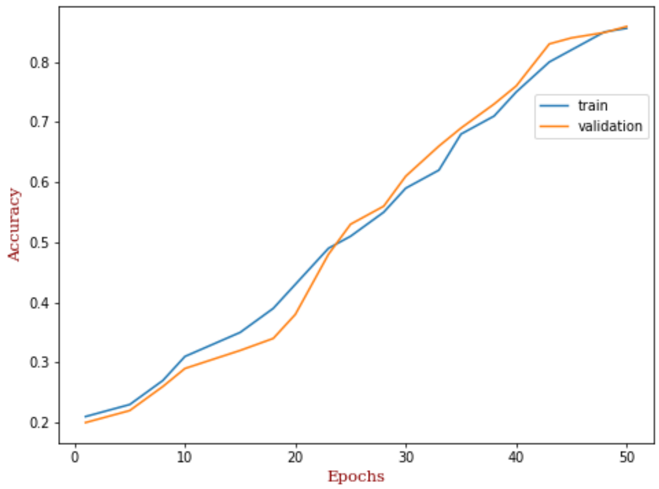


Figure 4.26: Accuracy of the RNN model for mapping keys with side-band

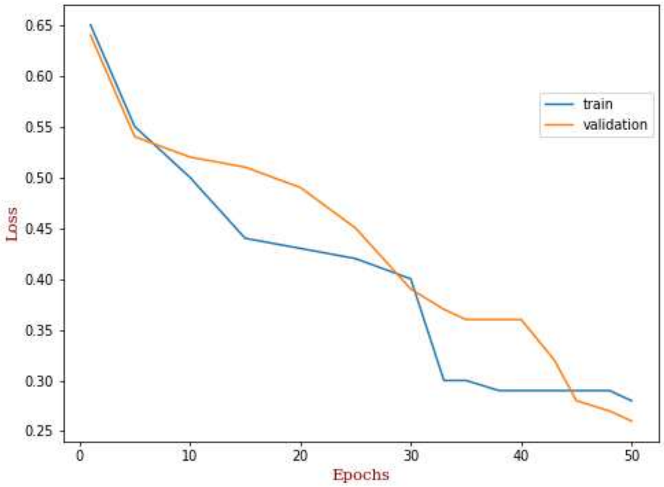


Figure 4.27: Loss of the RNN model for Mapping Keys with Side-band

Figure 4.28 shows the comparative performance of two RNN models in mapping the keys for ciphertext-plaintext pairs. We can see that the RNN model which use side-band information achieves better accuracy than the other RNN model. It is clear that using more information makes the models extract more features from the datasets and to achieve better results.

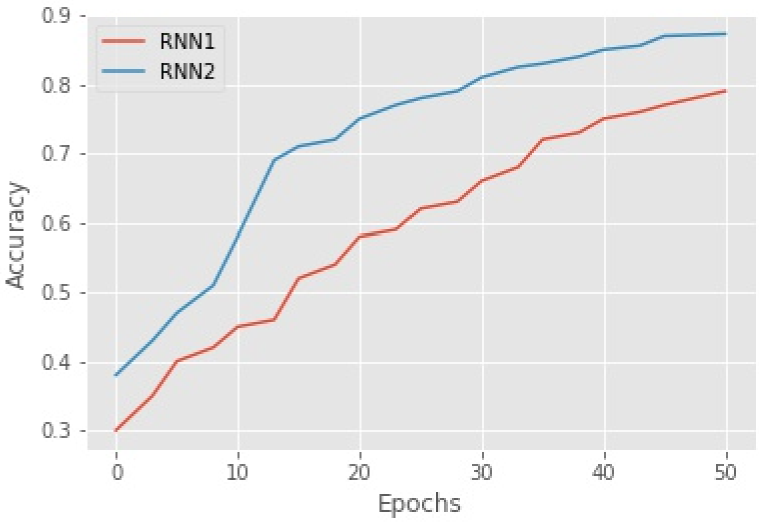


Figure 4.28: Comparison of two RNN models

From Table 4.6, we can see the relative difference between the performance of RNN models without and with side-band information.

Result Comparison III				
Model	Speed (Time per epoch)	Training Accuracy	Validation Accuracy	Loss
RNN1	125 seconds	78%	79%	23%
RNN2	160 seconds	87%	87.3%	19%

Table 4.6: Comparison of RNN & RNN on Key Mapping

CHAPTER 5

CONCLUSION

In this section, we will discuss the summary of our work and also suggest some directions for possible future work.

5.1 Overview of Our Work and Contributions

In our thesis work, we have studied side-channel attacks using deep learning techniques-more specifically, with two popular neural networks (CNN and RNN). We have discussed and pointed out that side-channel attacks are a real threat for the security and privacy of the devices and computing systems we use every day. We have shown how an attacker can extract valuable information from the computing devices like smart card readers, IoT devices, micro-controllers etc. using deep learning techniques. We have evaluated the performance of our models in predicting the DES rounds from the DPA contest dataset and toggle quantity count dataset. Using CNN and RNN models, we have showed how an attacker can detect the underlying algorithm from the datasets by simple classification. We have evaluated the performance of the two models in classifying between DES and Non-DES algorithms using various performance metrics. Finally, we have showed how an attacker can map key value for a specific plaintext-ciphertext pair. We showed that by using additional information (side-band), an attacker can get good accuracy in mapping keys.

5.2 Directions of Future Work

We have faced some challenges while working in this area of research.

1. Handling large datasets, we faced problems. Our machine became slow while processing these large datasets. We have used DPA contest V1 datasets and simulated toggle quantity datasets.

2. Processing the datasets to remove NULL or empty field. It took time and space to process large datasets.

3. Converting the input datasets into appropriate formats for applying the deep learning model.

4. We don't have adequate prior knowledge about deep learning techniques because we did not take any courses on them.

5. We couldn't build an automated process or scripts to find an exact neural network model by changing the hyper-parameters.

To address these, we worked hard and learned about deep learning techniques. We tried to complete our work with the resources that we already have. In the future, to process large datasets, we will occupy a server machine with high configuration. We will generate physical data from Artix-7 board and will conduct research. We will build a neural network model that will be very effective and will achieve higher accuracy.

REFERENCES

[1] Moradi A., Kasper M., Paar C. (2012) Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures. In: Dunkelman O. (eds) Topics in Cryptology CT-RSA 2012. CT-RSA 2012. Lecture Notes in Computer Science, vol 7178.

[2] Eisenbarth T., Kasper T., Moradi A., Paar C., Salmasizadeh M., Shalmani M.T.M. (2008) On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In: Wagner D. (eds) Advances in Cryptology CRYPTO 2008. CRYPTO 2008. Lecture Notes in Computer Science, vol 5157.

[3] Oswald D., Richter B., Paar C. (2013) Side-Channel Attacks on the Yubikey 2 One-Time Password Generator. In: Stolfo S.J., Stavrou A., Wright C.V. (eds) Research in Attacks, Intrusions, and Defenses. RAID 2013. Lecture Notes in Computer Science, vol 8145.

[4] Odelu, Vanga, Ashok Kumar Das, and Adrijit Goswami. ”A secure biometrics-based multi-server authentication protocol using smart cards.” IEEE Transactions on Information Forensics and Security 10.9 (2015): 1953-1966.

[5] Cagli, Eleonora & Dumas, Ccile & Prouff, Emmanuel. (2017). Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures. 45-68. 10.1007/978-3-319-66787-4_3.

[6] Samiotis, Ioannis Petros. Side-Channel Attacks using Convolutional Neural Networks. (2018).

[7] Maghrebi, Houssem et al. Breaking Cryptographic Implementations Using Deep Learning Techniques. IACR Cryptology ePrint Archive 2016 (2016): 921.

[8] Samiotis, I.P., 2018. Side-Channel Attacks using Convolutional Neural Networks: A Study on the performance of Convolutional Neural Networks on side-channel data.

[9] Kim, Jaehun et al. Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis. IACR Cryptology ePrint Archive 2018 (2018): 1023.

[10] Xie, X., 2002. Dynamics and learning in recurrent neural networks (Doctoral dissertation, Massachusetts Institute of Technology).

[11] Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I. and Vandewalle, J., 2011. Machine learning in side-channel analysis: a first study. Journal of Cryptographic Engineering, 1(4), p.293.

[12] Li, X., Qin, T., Yang, J. and Liu, T.Y., 2016. LightRNN: Memory and computation-efficient recurrent neural networks. In Advances in Neural Information Processing Systems (pp. 4385-4393).

[13] Bhandare, A., Bhide, M., Gokhale, P. and Chandavarkar, R., 2016. Applications of convolutional neural networks. International Journal of Computer Science and Information Technologies, 7(5), pp.2206-2215.

[14] Gradient descent basic idea `https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html`

[15] Convolutional Neural Network Demonstration `https://www.researchgate.net/figure/Structure-of-a-typical-convolutional-neural-network-CNN_fig3_323938336/`

[16] Keras Documentation `https://keras.io/`

[17] Tensorflow Documentation `https://www.tensorflow.org/`

[18] DPA Contest Website `http://www.dpacontest.org/home/`

[19] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. J. Wiener, editor, Advances in Cryptology - CRYPTO 99, 19th Annual International Cryptology Conference. Proceedings, volume 1666 of Lecture Notes in Computer Science, pages 388397. Springer, 1999.

[20] Deep Learning Applications `https://medium.com/breathe-publication/top-15-deep-learning-applications-that-will-rule-the-world-in-2018-and-beyond`

APPENDIX A

CORRELATION POWER ANALYSIS (CPA)

A.1 Overview

CPA is an attack that uses statistical analysis techniques and mathematical correlations, e.g. Pearson correlation coefficient. First, an intermediate small subkey is selected from the plaintext and part of the key. Then, the power consumption is hypothetically calculated for each essential subkey based on a power model (e.g. hamming weight model) for each plaintext sample. The Pearson correlation coefficient is calculated between the hypothetical power consumption values and the real power values measured. The key with the highest correlation with the measured trace is selected as the correct key guess.

CPA can be prevented by increasing the noise or decreasing the signal sizes, inserting timing variations, data masking, computing the inverse operations, changing the order of the computer operations by shuffling

A.2 Key Guess using Python Program

```
#!/usr/bin/env python


import sys


class CPA():


    sboxTable = ( 0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01,0x67,
 0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2,
0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64,
    )


    def __init__(self):
        pass

    def Sbox(self, inp):
        return self.sboxTable[inp]
        pass

    def HammingWeight(self, num, ret=0):

        count = 0

        while num >= 1:

            if num%2 > 0:
                count+=1
            num /= 2

        if ret == 0:
            return 6 - count
        else:
            return count

    def HammingDistance(self, num1, num2):

        return self.HammingWeight(num1^num2)
```

A.3 Recurrent Neural Network

RNN has different models:

- One to One: Takes one input and generates only one output, e.g. image classification.
- One to Many: Takes one input and generates a sequence of multiple steps, e.g. image captioning.
- Many to One: Takes a sequence of multiple steps as input and generates only one output, e.g. sentiment analysis.
- Many to Many: Takes a sequence of multiple steps as input and generates a sequence of words, e.g. language translations.

In this thesis work , we are using the many to one model where we are feeding the sequence of multiple toggle quantities of the power consumption from the Verilog HDL simulator into the RNN model and trying to achieve a single output if it is the desired encryption algorithm or not, and we are mapping many to many model to reverse engineer the encryption algorithm using the simulated toggle data.

APPENDIX B

POWER MEASUREMENT SETUP FOR XILINX ARTIX-7 BOARD

B.1 Overview

A post-silicon dataset refers to the dataset collected from the hardware implementation in actual devices at real-time.

This dataset was acquired by measuring the power consumption via probes from the Cmod A7 DIP form development board built around a Xilinx Artix-7 FPGA, which is considered the cryptographic device containing the code of the S-AES algorithm. Figure B.1 shows a Cmod A7 development board.

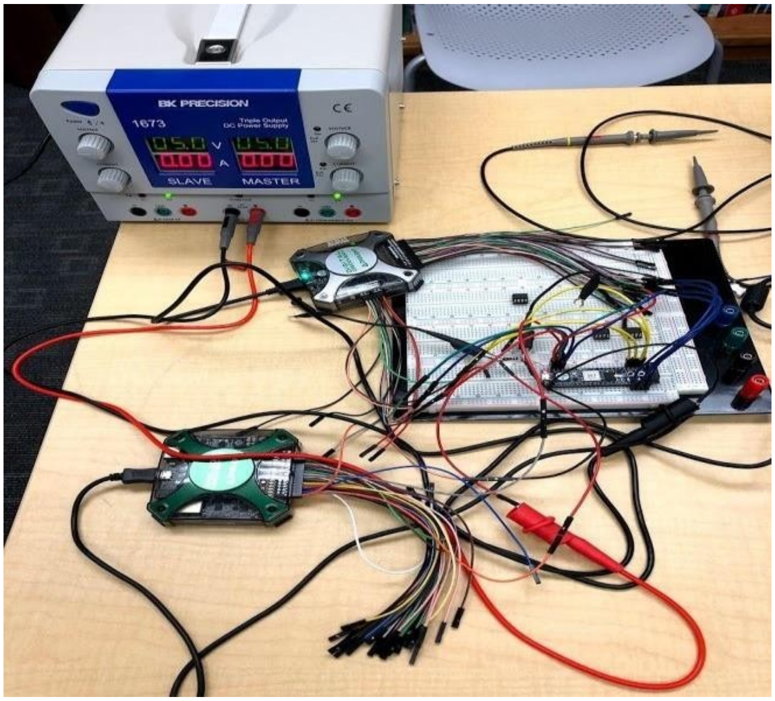


Figure B.1: Experimental Setup for Collecting Physical Dataset