Machine Learning(ML) based Side-Channel Analysis on AES Cryptosystem Through Power Consumption Data

A Final Evaluation Report

Master of Technology in Cryptology and Security

by

Sourin Manna

(Roll No. CrS2218)

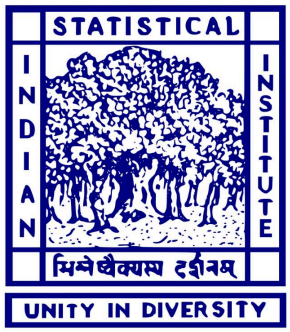
Under the Supervision of

Surya Prakash Mishra

DRDO, SAG Lab

Debrup Chakraborty

Associate Professor, ISI Kolkata



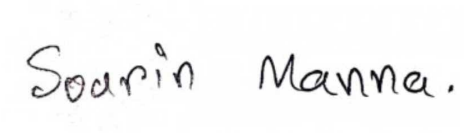
Indian Statistical Institute

Kolkata-700108,India

Jun 2024

Declaration

I, **Sourin Manna (Roll No: CrS2218)**, hereby declare that, this report entitled **"ML based SCA on AES Cryptosystem Through Power Consumption Data"** submitted to Indian Statistical Institute, Kolkata towards the fulfilment of the requirements for the degree of **Master of Technology** in **Cryptography & Security**, is an original work carried out by me under the supervision of **Surya Prakash Mishra and Debrup Chakraborty** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold academic ethics and honesty. Whenever a piece of external information or statement or result is used then, that has been duly acknowledged and cited.



Kolkata - 700 108 **Sourin Manna**

Date: .....

Place: .....

Certificate

This is to certify that the report entitled **"ML based SCA on AES Cryptosystem Through Power Consumption Data"**, submitted by **Sourin Manna** to the Indian Statistical Institute, Kolkata, for the award of the degree of **Master of Technology in Cryptology and Security**, is a record of the original, bona fide research work carried out by him under our supervision and guidance. The report has reached the standards fulfilling the requirements of the regulations related to the award of the degree.

The results contained in this report have not been submitted in part or in full to any other University or Institute for the award of any degree or diploma to the best of our knowledge.

.....

Surya Prakash Mishra
Side Channel Cryptanalysis, SAG Lab,
Defence Research and Development Organisation.

.....

Debrup Chakraborty
Department of Cryptology and Security,
Indian Statistical Institute, Kolkata.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Surya Prakash Mishra, DRDO & Debrup Chakraborty, ISIK , for their invaluable guidance, support, and encouragement throughout this project. Their expertise and insights have been instrumental in shaping the direction of this study.

I am also grateful to Indian Statistical Institute, Kolkata for providing the necessary resources and facilities to conduct this research.

Souri Manna

Abstract

This Internship Project explores the vulnerabilities of the Advanced Encryption Standard (AES) cryptographic system to side-channel attacks, specifically focusing on leveraging power consumption data. We know AES is a secure encryption crypto system and we try to break the security by a method called side-channel analysis(SCA), where hackers can figure out the encryption key by watching how much power a device uses when it's doing the encryption. Our goal is to find the key for this encryption. We applied a different machine learning model initially for better comprehension we took the model whose accuracy score was better, suitable for anyone with basic knowledge of machine learning. If we succeed in our analysis, we will identify any weaknesses in my encryption, address the issues, and explore the development of a more robust model in the future. We aim to predict a 128-bit key using a multiclass classification approach. By utilizing known plaintext and power consumption data, our objective is to conduct an attack on AES encryption. We trained a model with power consumption data as independent features, where the dependent feature is the Hamming weight of the string obtained after performing XOR between the plaintext and key, followed by passing through the S-box. To acquire dependent data (or labeled data) for training purposes, we provided the key.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The motivation behind this research stems from the increasing need to secure cryptographic systems against sophisticated attacks. Despite the robustness of AES, the emergence of side-channel attacks poses a significant threat to its security. Traditional cryptanalysis focuses on the mathematical aspects of cryptographic algorithms, but side-channel attacks exploit physical properties, such as power consumption, to extract secret keys.

Given the proliferation of internet-connected devices and the critical importance of securing communications, understanding and mitigating side-channel vulnerabilities is crucial. Machine learning has shown promise in various domains, and its application to side-channel analysis represents a novel and potentially powerful approach to enhancing cryptographic security.(Jack Edmonds, 2021)

1.2 Problem Statement

Give large amount of data we apply various Machine Learning algorithm to analyze the vulnerability of the AES cryptosystem applied to power consumption data, aiming to predict encryption key information from observed power patterns during encryption. Especially in this problem, we have 20,000 rows and 17,000 columns of independent data. The dependent data is the Hamming weight, which we obtain after performing AES. Then we apply machine learning algorithm to predict hamming weight.

1.3 Study Object

The primary objective of this study is to investigate the vulnerabilities of the AES cryptosystem to power-based side-channel attacks using machine learning techniques. Specifically, this study aims to:

1. Identify potential weaknesses in the AES cryptosystem that can be exploited through side-channel attacks.

2. Develop and evaluate machine learning models that can accurately predict the Hamming weight of the ciphertext from power consumption data.

3. Demonstrate the feasibility of using machine learning for side-channel analysis and highlight the importance of implementing countermeasures to protect against such attacks.

1.4 Power Base Attack

Like other side-channel attack Timing Attacks,Cache Attacks,Fault Injection Attacks.Power Base Attacks be more effective, more practical to mount and cheaper to install (with equipment such as power sensors or oscilloscopes) than other side channels.Power Base Attacks are a form of side-channel attack that exploit the power consumption patterns of cryptographic devices to extract secret keys and sensitive data. By measuring and analyzing the power traces during cryptographic operations, attackers can correlate these patterns with the underlying computations. This form of attack highlights the vulnerabilities in hardware implementations of cryptographic algorithms, necessitating advanced countermeasures to ensure robust security.

1.5 Data Understanding

1.5.1 Hamming Weight

- The Hamming weight of a string is the number of symbols that are different from the zero-symbol of the alphabet used. It is most commonly applied to binary strings (sequences of bits) where the zero-symbol is 0 and the non-zero symbol is 1. In this context, the Hamming weight is simply the number of 1s in the binary string.For example, The Hamming weight of the binary string 1101 is 3, because there are three 1s in the string.

The power consumption dataset comprises 20,000 rows, representing the power consumption during the encryption of 20,000 plaintexts using AES, and 17,000 columns, signifying 17,000 features. These features are gathered by measuring power consumption every nanosecond. The dataset consists of 20,000 rows and 17,000 columns,

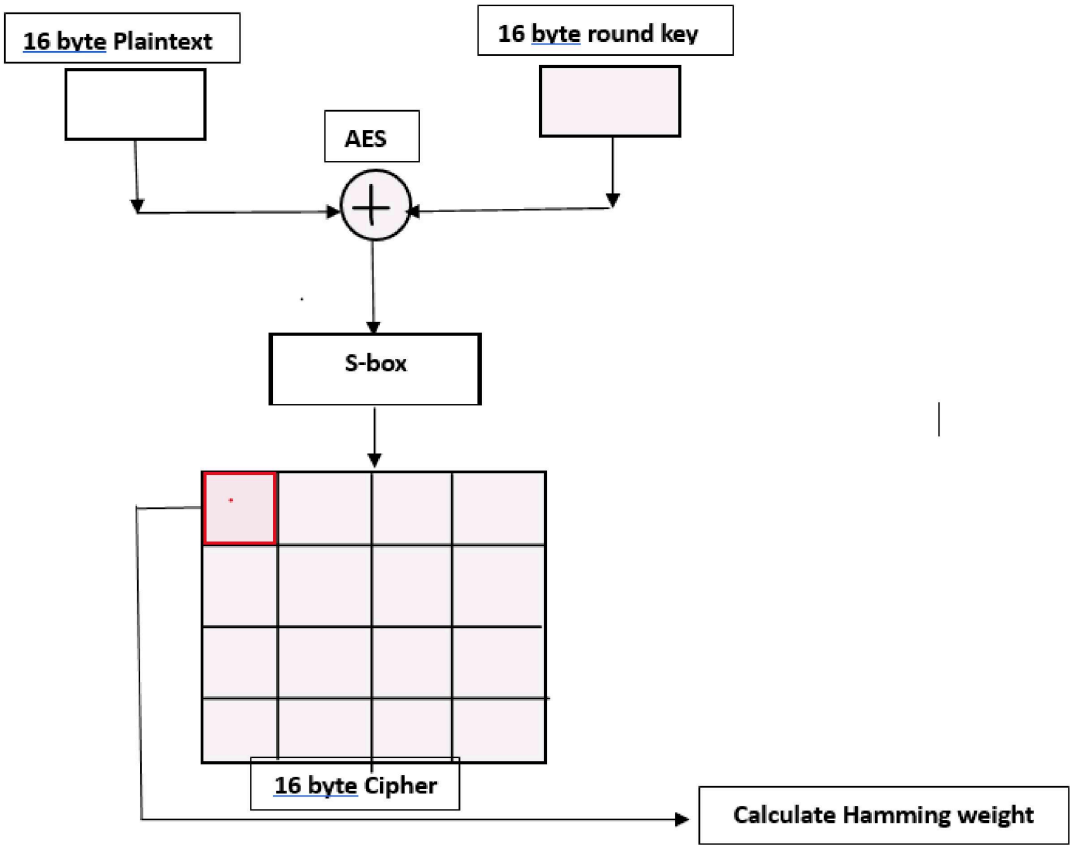


FIGURE 1.1: Diagram of Calculating dependent variable or hamming weight

representing our independent data. Now, we need to determine the dependent data, which is the Hamming weight. To achieve this, we perform the first two steps of the AES algorithm. Initially, we have the plaintext and the key. First, we perform an XOR operation between the plaintext and the key to obtain a binary string of 0s and 1s. Next, we pass this binary string through the S-box. Finally, we calculate the Hamming weight of the first 8 bits of the resulting string, which serves as our dependent variable, which ranges between 0 to 8.(Stjepan Picek, 09 February 2023)

Now, we aim to understand the power consumption values related to the first feature or the first nanosecond as analyzed by a cryptographic algorithm.

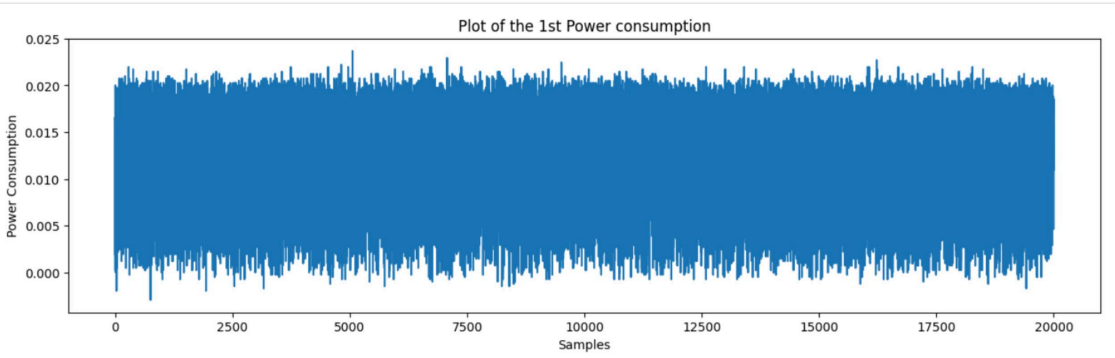


FIGURE 1.2: Power Consumption for 1st sample of given data set

1.6 Summary

In summary, this study aims to explore the vulnerabilities of the AES cryptosystem to side-channel attacks using machine learning techniques. By analyzing power consumption data, we aim to identify potential weaknesses and develop models that can accurately predict sensitive information. This research highlights the importance of understanding and mitigating side-channel vulnerabilities to ensure robust cryptographic security.

Chapter 2

Background

2.1 Side-Channel Analysis

One method of attacking cryptographic algorithms by attempting to exploit hardware vulnerabilities is side-channel analysis (SCA), which involves analyzing device information associated with the internal workings of an algorithm. More specifically, side-channel analysis entails learning information about a system by analyzing things like device power consumption, electromagnetic radiation, timing, and sound, and in doing so attempting to reveal information that was intended to remain unknown to unauthorized parties

2.2 Machine Learning

Machine learning is an area of computer science a subset of Artificial Intelligence(AI) that uses algorithms to learn from data to build computer systems that can learn and

improve on their own. In our project, we used different types of machine-learning classification algorithms:(Osisanwo F.Y., 3 June 2017)

- **Logistic Regression:** An algorithm that uses a logistic model to calculate the probability of a result occurring and makes a prediction based on its probability. It models the relationship between the dependent binary variable and one or more independent variables using a logistic function, making it suitable for binary classification tasks.

- **Random Forest:** Random Forest is an ensemble of decision trees for classification and regression tasks. It constructs multiple decision trees during training and outputs the majority vote of these trees, improving accuracy and controlling overfitting through randomness in data sampling and feature selection.

- **Naive Bayes:** Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming independence among features. It calculates the posterior probability for each class and predicts the class with the highest posterior probability, making it effective for large datasets and text classification tasks.

- **Support Vector Machine(SVM):**This algorithm distinguishes between two or more classes within the dataset by using a linear combination of the features. The way it optimizes this is by using the points of different classes that are closest to each other as support vectors and then making a linear boundary at the center of those support vectors.

- **Artificial Neural Network(ANN):** Artificial Neural Network mimics the human brain's neural network structure to learn complex patterns. It consists of layers of interconnected nodes (neurons) that adjust their weights during

training to minimize prediction error, making ANNs suitable for capturing non-linear relationships in data.

- **K-nearest neighbor:** K-nearest neighbor assigns a class to an unlabeled example based on the majority class of its k nearest neighbors. It involves calculating the distance between the new input and existing points, making it simple to implement but computationally expensive for large datasets.

- **XGBoost:** XGBoost is a gradient boosting to iteratively build a series of decision trees for predictive modeling. By focusing on correcting errors in previous trees, it enhances model performance with techniques like parallel processing and regularization to avoid overfitting.

- **Multilayer Perceptron Network:** A model with a neural network architecture where every node in one layer connects to every node in the previous and next layer. The activation function at each node is used to calculate the weights for each layer that influence the output

2.3 Advanced Encryption Standard (AES-128)

AES, or the Advanced Encryption Standard is a symmetric encryption algorithm(where the same key is used for both encryption and decryption) and a block cipher and is one of the most widely used symmetric-key algorithms today.AES can operate with 128, 192, or 256-bit keys, which is a significant step up from what is commonly regarded as its predecessor.AES became a replacement for DES because the effective key size of 56 bits for DES eventually became too susceptible to brute force (or exhaustive search) attacks as computer processing power improved. The smallest AES

key size offers 2^{128} = 3.4e+38 possible combinations, which is considerably larger than the 2^{56} = 7.2e+16 key space offered by DES. Here we work with 128-bit AES.

AES is a block cipher, which means that chunks of bits are operated on together instead of each bit being dealt with separately. It operates on 16 byte-sized chunks (128 total bits) of data regardless of key size, and a series of operations is performed on the 4x4 state array of this data a certain number of times depending on the key size. AES-128 has 10 rounds, AES-192 has 12, and AES-256 has 14. AES-128 was the variant we focused on, and its general flow of encryption operations can be seen in Figure 2.1.

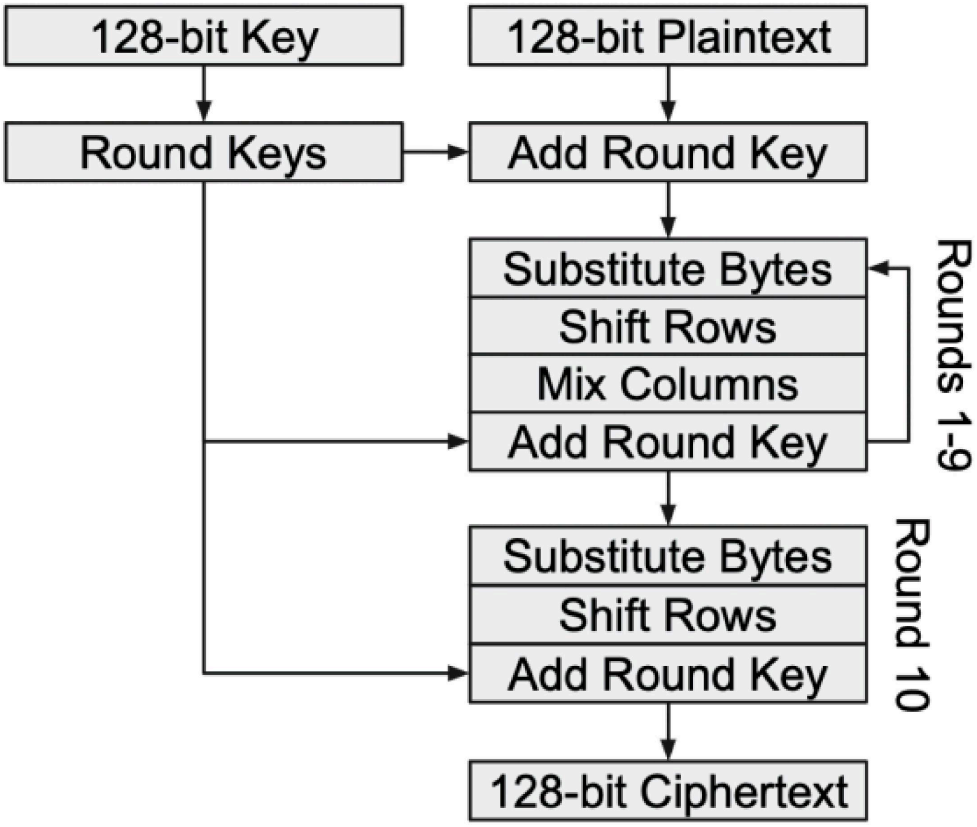


FIGURE 2.1: AES-128 encryption flow (decryption is a similar process with inverse operations)

The "add round key" operation in symmetric-key cryptography involves taking the XOR of the state array bytes and the corresponding round key bytes. This invertible XOR operation allows for encryption and decryption, as demonstrated by the example where 1100 (plaintext) XOR 0101 (key) results in 1001 (ciphertext), and 1001 (ciphertext) XOR 0101 (key) returns to the plaintext 1100.

The "substitute bytes" or S-box operation in AES replaces each of the 16 bytes in the state array with a value from the S-box lookup table, which contains 256 possible values (0-255) used as indices for substitution. To protect against side-channel attacks (SCA), one AES-128 implementation we evaluated applied masking by XORing each S-box output byte with an additional mask value, disrupting the first-order relationship between power consumption spikes and S-box output byte values, with the mask being invertible and removable if its value is known.

The "shift rows" operation in AES applies circular left shifts to the rows of the state array: the first row is unchanged, the second row shifts left once, the third row shifts left twice, and the fourth row shifts left three times. In a circular shift, the leftmost value of a row moves to the far right after the shift.

The "mix columns" operation involves taking the matrix-vector product of a Maximum Distance Separable (MDS) circulant matrix and each column of the state array. This operation, which is excluded from the final round to avoid redundancy in security enhancement, is designed to obscure the relationship between plaintext and ciphertext.(Donald L. Evans,Phillip J. Bond,Karen H. Brown, 2001)

Chapter 3

Study Methodology

3.1 Introduction

In this study, we analyze the power consumption of the AES encryption process to predict the Hamming weight of the first 8-bit of ciphertext. The Hamming weight(dependent variable) is obtained after performing an XOR operation between the plaintext and the key, followed by passing the result through the S-box. We leveraging machine learning models, we aim to predict this Hamming weight based on the power consumption data(independent variables). This approach is inspired by side-channel analysis techniques, which exploit physical leakages during cryptographic operations.

3.2 Experimental Setup

3.2.1 Dataset Description:

- **Independent Data:** The dataset consists of 20,000 rows and 17,000 columns. Each row represents the power consumption recorded during the encryption of a plaintext using AES. The 17,000 columns represent the power consumption measurements taken at nanosecond intervals. These measurements capture the subtle variations in power usage that can reveal information about the encryption process.
- **Dependent Data:** The target variable is the Hamming weight of the first 8-bit of the ciphertext. This is calculated using the following cryptographic steps:
    1. Perform an XOR operation between the plaintext and the AES key.
    2. Perform an XOR operation between the plaintext and the AES key.
    3. Calculate the Hamming weight of the resulting 8-bit value. **The power consumed will be linearly dependent on the Hamming Weight of data being processed.**

3.2.2 Data Collection:

Power consumption data was collected using a high-resolution oscilloscope, which measures the power usage of the cryptographic device at nanosecond intervals during the AES encryption process. This data helps in identifying specific power consumption patterns associated with different cryptographic operations.

3.2.3 Data Preprocessing:

1. **Normalization Independent Feature:** We normalize the power consumption values to ensure uniform scaling across all features. Here we need data normalization because of data security, we can not give raw data to our machine learning model.

2. **Data Integration and Segmentation:** We have two tables: one containing the independent data (power consumption data) and one containing the dependent data (Hamming weight label data). First, we merge the two tables with respect to the plaintext. Then, we split the entire dataset into training and testing parts.

3.3 Evaluation Process

3.3.1 Cryptographic Operation Extraction:

To derive the dependent variable, we perform the following operations:

- **XOR Operation:** The plaintext byte is XORed with the key byte to produce an intermediate value. This operation is fundamental in AES encryption as it combines the plaintext with the key in a simple yet effective manner.

- **S-box Transformation:** The intermediate value is then passed through the AES S-box, a non-linear substitution step. The S-box introduces non-linearity and confusion, making it harder for attackers to reverse-engineer the encryption process.

- **Hamming Weight Calculation:** The Hamming weight of the S-box output (first 8-bit) is calculated, serving as the target for our prediction models. The Hamming weight represents the number of bits set to '1' in the byte, a metric often used in side-channel analysis due to its correlation with power consumption.

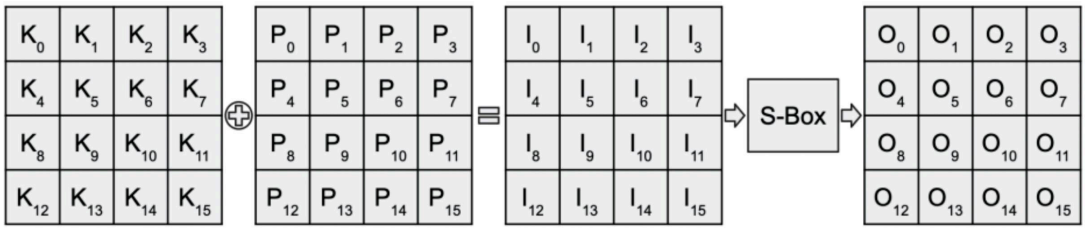


FIGURE 3.1: AES-128 encryption flow (Flow of the "add round key" and "substitute bytes" operations in the first round of AES-128)

3.3.2 Model Selection:

We apply several multi-class machine learning models to predict the Hamming weight. The chosen models include:

- Logistic Regression: A baseline model for multi-class classification.
- XGBoost: Is a ensemble learning using sequential processing.
- Random Forests: An ensemble method to improve the robustness and accuracy.
- Neural Networks: Deep learning models to capture complex patterns.

3.3.3 Training and Testing:

We are given a table with 20,000 rows and 17,001 columns. We will now apply various machine learning algorithms and determine the accuracy of each model.

1. **Data Split:** The dataset is split into training (80%) and testing (20%) sets to evaluate the generalization performance of the models. This split ensures that the models are tested on unseen data, simulating real-world scenarios.

2. **Model Training:** Each model is trained on the training set. Hyperparameters are tuned using cross-validation to prevent overfitting and ensure optimal performance. Cross-validation helps in validating the model's performance across different subsets of the data.

3. **Model Testing:** The trained models are then evaluated on the testing set. This evaluation phase assesses how well the models can predict the Hamming weight on new, unseen data.

3.3.4 Performance Metrics:

Basically we evaluate our all the models using the following metrics:

- **Accuracy:** The ratio of correctly predicted instances to the total instances. It provides a basic measure of the model's overall performance.

- **Precision:** The percentage of correctly predicted positive cases out of all predicted positive cases.

- **Recall:** The percentage of correctly predicted positive cases out of all actual positive cases.

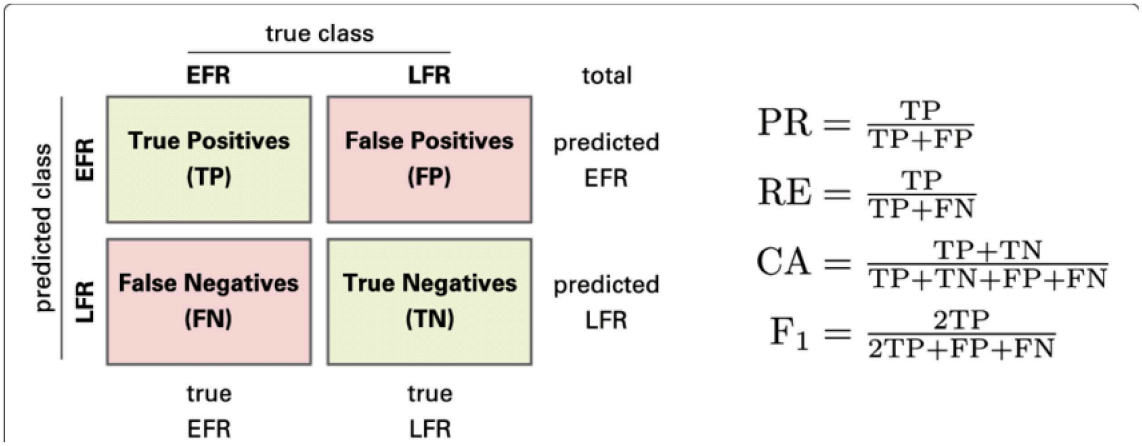


FIGURE 3.2: Confusion matrix. Exemplified CM with the formulas of precision (PR), recall (RE), accuracy (CA), and F 1-measure

- **Confusion Matrix:** To visualize the performance and identify misclassifications. The confusion matrix offers insights into which Hamming weights are frequently mispredicted.

Chapter 4

Analysis and Results

4.1 Introduction

Our rigorous experimentation involved various machine learning models to achieve this goal. Among the models tested, XgBoost emerged as the most effective, achieving a significant accuracy rate of 63.9% in predicting the hamming weight. This result outperformed Logistic Regression (44.2%), Random Forest (47.9%), and Artificial Neural Networks (ANN) (43.95%). These findings highlight the promising role of advanced machine learning techniques in exposing vulnerabilities within AES, paving the way for further research to bolster encryption security.

4.2 Discussion

Our investigation involved applying various machine learning models to predict the hamming weight based on power consumption traces. The models evaluated include XgBoost, Logistic Regression, Random Forest, and Artificial Neural Networks

(ANN). Each model's performance was assessed based on its accuracy in predicting the hamming weight.

1. **Logistic Regression:** As a baseline model, Logistic Regression achieved an accuracy of 44.2%. Although it provides a straightforward approach to classification problems, its performance was limited in this context due to the complexity of the data and the non-linear relationships inherent in side-channel information.

2. **Artificial Neural Networks (ANN):** With the potential to model complex non-linear relationships, ANN achieved an accuracy of 43.95%. However, its performance was hindered by the limited size of the dataset and the need for extensive hyperparameter tuning and computational resources.

3. **XgBoost:** This gradient boosting framework demonstrated the highest accuracy of 63.9%. XgBoost's capability to optimize performance through boosting, regularization, and handling of missing data likely contributed to its superior performance in predicting the hamming weight. The model's success underscores the effectiveness of advanced machine learning techniques in uncovering vulnerabilities in cryptographic systems.

4.3 Results

The table below summarizes the performance of each machine learning model in predicting the hamming weight of the first 8 bits of the ciphertext after the XOR operation with the key and S-box transformation.

Model	Accuracy (%)	Precision(%)	Recall(%)
Logistic Regression	44.2	0 or 44.1	0 or 100
Random Forest	47.9	53.4	76
Naive Bayes	29.6	67.8	33.7
Artificial Neural Networks	43.95	-	-
K-nearest Neighbour	32.9	-	-
XgBoost	63.9	-	-

TABLE 4.1: Performance metrics of machine learning models in predicting the hamming weight of the first 8 bits of the ciphertext.

```
Accuracy: 0.44233333333333336
Classification Report:
              precision    recall

           2       0.00      0.00
           3       0.00      0.00
           4       0.44      1.00
           5       0.00      0.00
```

FIGURE 4.1: Classification Report of Logistic Regression model of four classes 2,3,4, and 5.

4.4 Conclusion

Our findings indicate that machine learning models, particularly XgBoost, have significant potential in identifying vulnerabilities in AES encryption systems through side-channel attacks. The higher accuracy achieved by XgBoost demonstrates the value of advanced machine learning techniques in this domain. Future research can focus on optimizing these models further, exploring additional side-channel data, and developing more robust defenses against such attacks to enhance encryption security.

In our analysis, we focused on the Hamming weight of the first 8 bits of the ciphertext after XOR with the key and S-box transformation. Suppose we could build a

multiclass classification machine learning model with 128 classes. In this model, the Hamming weight would serve as the independent variable, and the output would be the actual 128-bit key, represented as 128 classes (the dependent variable). This approach could allow us to predict the original key used in AES encryption.

Alternatively, If we can establish any relationship between the original key and the Hamming weight, or if we calculate the inverse of the S-box and key expansion algorithm, we may be able to determine the original key.

Chapter 5

Conclusions and Further Work

5.1 Summary

With hardware security attacks being prevalent in our digital age, it's crucial to identify vulnerabilities in cryptosystems used to protect sensitive information. In line with this objective, the goal of our final semester project was to evaluate the machine learning-based side-channel attack resistance of the AES-128 cryptosystem using power consumption data. Our focus was on predicting the Hamming weight of the first 8-bit of ciphertext obtained after the XOR operation between the plaintext and key, followed by passing through the S-box.

We collected power consumption data during the encryption process, using a dataset comprising 20,000 plaintexts and 17,000 power consumption features measured at nanosecond intervals. Utilizing a side-channel leakage model targeting the first round S-box operation of AES, we applied several machine learning classifiers to predict the Hamming weight.

In our evaluation, we discovered that the AES-128 implementation is vulnerable to this form of attack. However, masking techniques can mitigate this vulnerability when the attacked key differs from the profiled key. Among the classifiers we tested, we found that while all models were capable of predicting the Hamming weight, some required fewer power consumption traces to achieve accurate predictions than others. Specifically, our results showed that XgBoost outperformed other classifiers, achieving a 63.9% accuracy rate, compared to Logistic Regression (44.2%), Random Forest (47.9%), and ANN (43.95%).

We can predict the AES key by using a 128-class machine learning model, with Hamming weight as input and the key as output.

5.2 Further Works

Currently I am working with only 1st round AES-128 next we plan to work with more round AES-128. After that, we plan to extend our research to 192 and 256-bit AES encryption keys, exploring their vulnerabilities to side-channel attacks. Additionally, we aim to analyze other side-channel data, such as Timing Attacks, Cache Attacks, and Fault Injection Attacks. We will also investigate the application of more advanced machine learning/deep learning algorithms, including Generative Adversarial Networks (GANs), to improve prediction accuracy and develop robust countermeasures against these sophisticated attacks.

References

Donald L. Evans,Phillip J. Bond,Karen H. Brown, 2001. Advanced Encryption Standard (AES) , 46.

Jack Edmonds, T.M., 2021. Machine learning-based side-channel analysis on the advanced encryption standard , 1–5.

Osisanwo F.Y., Akinsola J.E.T., A.O., 3 June 2017. Supervised machine learning algorithms. IJCTT 48, 7–9.

Stjepan Picek, Guilherme Perin, L.M.L.W.L.B., 09 February 2023. Deep learning-based physical side-channel analysis. ACM Journals 55, 4–11.