ANALYZING AES POWER TRACES FOR SIDE-CHANNEL ATTACKS: GENERATION, CLASSIFICATION, KEY DEDUCTION, AND MITIGATION STRATEGIES

by

Keerthana Rajeev

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
July 2024

I dedicate this work to Lord Guruvayoorappan and my family.

Table of Contents

List of Tables

List of Figures

Abstract

Side-channel attacks (SCAs) exploit leakages from a system's physical implementation, like acoustic signals, electromagnetic, and power emissions, to deduce sensitive information, thus bypassing traditional security measures. SCAs appeal to hackers due to their non-invasive nature and low cost and thereby necessitate robust countermeasures.

The Advanced Encryption Standard (AES) is a widely used symmetric key cryptography known for its robustness, but it remains susceptible to SCAs. This research analyzes power traces to identify vulnerabilities, classifies power traces based on AES implementation, employs Deep Learning(DL) models to deduce cryptographic keys, and develops mitigation strategies against SCAs.

We generated and analyzed real and synthetic power traces from masked AES implementations using a Syscomp waveform generator and a Python script. Techniques like Fast Fourier Transform (FFT), Wavelet transform, and linear regression were used to correlate the traces. Power traces from the AES Power Trace (AES_PT) dataset were classified into three AES implementations using feature extraction techniques and Support Vector Machine (SVM) classification based on statistical properties from Principal component analysis (PCA).We used hashing and metadata techniques retrieved original power traces from the feature set.

The study used ANSSI Side-channel Analysis Database (ASCAD) and adopted deep learning models for key deduction: Residual Networks were transformed into ResTraceNet using 1D convolutional layers, and Gated Recurrent Units (GRUs) were modified into GRUTrace to process 1D power traces. These models deduced one key byte using only 100 power traces, achieving testing accuracies of 96.68% and 96.28%. We proposed a mitigation strategy involving structured masking and Gaussian noise to obscure relationships between cryptographic keys and power consumption patterns.

Our proposed research provides a comprehensive analysis of AES power traces, using DL models to perform feature extraction, classify and deduce cryptographic keys, and proposes mitigation techniques to enhance defenses against SCAs.

List of Abbreviations and Symbols Used

Acronyms

3DES	Triple DES
AESPT	AES Power trace
AES	Advanced Encryption Standard
ASCAD	ANSSI Side-channel Attack
AUC	Area Under Curve
B-CNN	Bilinear Convolutional Neural Networks
CCA	Cross-correlation Analysis
CER	Cross-Entropy Ratio
CNN	Convolutional Neural Network
CO-DLA	Correlation Optimization Deep Learning Analysis
CPA	Correlation Power Analysis
CV	Coefficient Of Variation
CWT	Continuous Wavelet Transform
DDLA	Deep Learning-based Differential Learning Analysis
DES	Data Encryption Standard
DL	Deep Learning
DNN	Deep Neural Network
DPA	Differential Power Analysis

DTW	Dynamic Time Warping
ECC	Elliptic Curve Cryptography
EDA	Estimation of distribution algorithms
ELM	Extreme Learning Machine
EM	Electromagnetic Emissions
FFT	Fast-Fourier Transform
FI	Feature Importance
FPR	False Positive Rate
GE	Guessing Entropy
GRU	Gated Recurrent Network
HD	Hamming Distance
HW	Hamming Weight
ICA	Independent Component Analysis
IQR	Interquartile Range
LASSO	Least Absolute Shrinkage and Selection Operator
LSTM	Long Short-Term Memory
MLP	Machine Learning
MLP	Multi-Layer Perceptron
MTD	Minimum Traces to Disclosure
NIST	National Institute of Standards and Technology
PA	Power Analysis

PCA	Principal Component Analysis
PI	Permutation Importance
Q1	First Quartile
Q3	Third Quartile
Relu	Rectified Linear Unit
ResNet	Residual Network
RFE	Recursive Feature Elimination
RF	Random Forest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RSA	Rivest, Shamir, and Adelman
SCA	Side-channel attack
SHAP	SHapley Additive exPlanations
SHA	Secure Hashing Algorithm
SPA	Simple Power Analysis
STD	Standard Deviation
SVM	Support Vector Machine
TA	Template Attack
TPR	True Positive Rate
WT	Wavelet Transform

Acknowledgements

First, I want to thank God and my family for their constant support and encouragement, which have been my guiding light throughout this journey. I am especially grateful to my parents, Rajeev and Sreejaya who have been with me every step of the way, offering wisdom and love, and to my sister, Anagha whose faith and belief in my abilities have continually inspired me to aim for the stars. Thank you for being my motivation and cheerleader. I also extend heartfelt thanks to my grandmothers, whose guidance and support have been crucial in my educational journey.

Secondly, I want to express my deepest gratitude to Dr. Srinivas Sampalli for his outstanding support. His guidance, encouragement, and unwavering belief in my abilities have been invaluable. Dr. Sampalli's expertise and dedication have profoundly influenced my academic and personal growth, and I am incredibly grateful for his mentorship.

I would also like to thank Dr. Darshana Upadhyay for her continuous support. Her regular check-ins and constructive feedback have been monumental in shaping my research journey and keeping me on the right track. Dr. Upadhyay's insights and encouragement are greatly appreciated.

Additionally, I am thankful to Dr. Jaume Manero for his technical expertise and support. His knowledge and assistance have been essential in navigating the crucial aspects of my research.

I am also immensely grateful to my best friends, Sushumna, Naveen, and Dheemanth, whose support has been invaluable. They provided not just guidance but also the joy and friendship that sustained me through this beautiful journey.

Lastly, I want to acknowledge my best friends back home, Akshay and Vaishakh, along with all my friends and family who have helped me immensely and everyone who has supported me in any way during this journey.

Chapter 1

Introduction

Side-channel attacks (SCAs) pose a significant threat to systems by targeting their physical implementation rather than exploiting flaws in the design of the algorithm or protocol. These attacks analyze various side-channel information such as electromagnetic emissions, power consumption patterns, acoustic signals, and thermal patterns to extract sensitive information[1]. In the context of cryptography, side-channel information leaked during the execution of a cryptographic operation, such as the S-box operation in the Advanced Encryption Standard (AES) algorithm, can be used to deduce sensitive information, including cryptographic keys or fragments of the cryptographic key (e.g., 1 or 2 bytes) [2]. Figure 1.1 shows a side-channel analysis.

These attacks were first discovered and introduced by cryptographer Paul Kocher in 1996, and over the years, they have gradually grown stronger. Kocher identified a timing attack that exploited variations in the timing of various cryptographic operations to extract sensitive information. [3].

SCAs can be broadly classified into two major categories based on: a) the nature of the leakage they exploit, and b) the methodology used to perform the attack. In the former category, the attack is carried out based on the type of side-channel used, where accidental leakages are utilized by the attackers. These leaks can arise from various sources such as timing information, electromagnetic emissions, power consumption, acoustic emissions, thermal emissions, and so on [4]. The crux of this research explores Power Analysis (PA), which monitors power consumption patterns. PA attacks are classified into three categories namely, Simple Power Analysis (SPA), Differential Power Analysis (DPA), and Correlation Power Analysis (CPA). SPA directly observes power variations to deduce information regarding cryptographic operations [5]. DPA is a powerful technique that involves collecting large amounts of power traces from a cryptographic device. CPA builds upon the capabilities of DPA by applying more refined and enhanced statistical techniques [6].

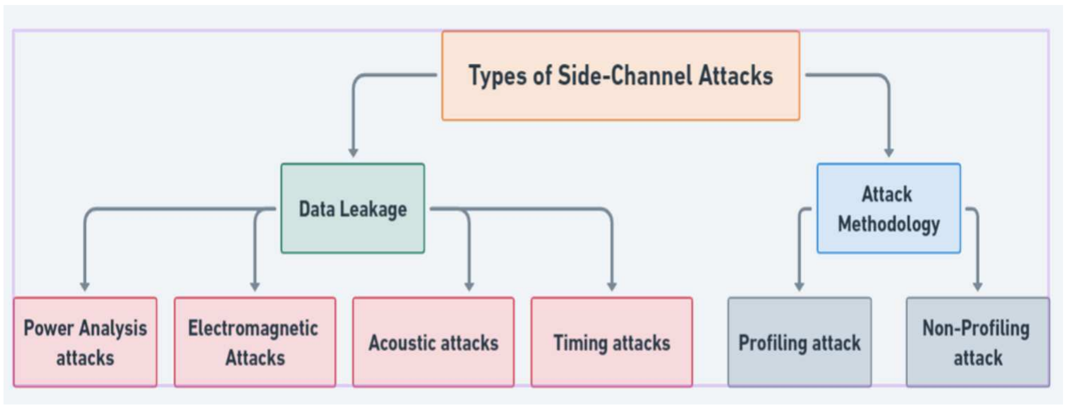


Figure 1.1: A high-level overview of side-channel attacks

The latter category of attacks relies on techniques and methods used to extract sensitive information from side-channels and is classified into profiling and non-profiling attacks. Profiling attacks involve creating a model of the target device under various conditions and using this model to interpret attack data. Examples include Template Attack (TA) and Deep Learning (DL)-based attacks [7]. In non-profiling attacks, there is no profiling phase; data is directly analyzed without prior processing. DPA and CPA fall into this category [8].

SCAs have significant implications for the security and integrity of cryptographic algorithms. The non-invasive nature of SCAs is particularly concerning, as most existing attack techniques focus on directly targeting the device or causing physical damage. SCAs can also bypass traditional defense mechanisms, which are primarily aimed at software vulnerabilities. The low cost and high effectiveness of SCAs make them extremely appealing to attackers, who can execute these attacks with limited resources and achieve substantial success. Failure to address these vulnerabilities can lead to severe consequences, including data breaches, financial loss, and erosion of user confidence in digital security solutions [9].

Both symmetric and asymmetric key cryptographic algorithms are vulnerable to SCAs. Some of the most widely known algorithms include symmetric key cryptographic algorithms like the AES and the Data Encryption Standard (DES), as well as asymmetric key cryptographic algorithms such as RSA (Rivest-Shamir-Adleman) and Elliptic Curve Cryptography (ECC). Regardless of the cryptographic properties

AES possesses, SCAs can be easily executed because these attacks exploit the physical implementation of AES, particularly by observing power trace profiles to infer the encryption stages or deduce keys.[10].

Traditional methods like DPA and CPA face significant challenges. They require vast amounts of power traces and expert domain knowledge, and are highly susceptible to noise and jitters [11] These methods assume linear relationships between power consumption and cryptographic operations, missing intricate interactions. Additionally, DPA and CPA struggle with scalability, compromising performance and efficiency with high-dimensional and large-volume data [12].

Deep Learning (DL) offers good solutions to the limitations of traditional methods. DL models can autonomously perform feature extraction from raw power traces, identifying necessary features without expert input. They detect complex non-linear relationships between side-channel information and sensitive data, providing a comprehensive attack overview. DL models are highly accurate, robust against noise, and require fewer traces for information deduction when properly trained [13]. Additionally, they can generalize from large datasets, handling a wide range of cryptographic algorithms and attack scenarios. DL models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can analyze temporal and spatial patterns in power traces, creating models powerful enough to break strong cryptographic algorithms [14].

We focus on implementing two robust DL models, ResTraceNet (adapted from Residual Networks, originally based on CNN architectures) and GRUTrace (adapted from Gated Recurrent Units), to demonstrate the effectiveness of side-channel attacks (SCAs). ResTraceNet adapts the deep CNN-based architecture of Residual Networks for sequential data analysis, making it well-suited for cryptographic settings [15]. Meanwhile, GRUTrace, evolved from traditional GRU components of Recurrent Neural Networks (RNNs), excels at modeling dependencies in sequential data, making it particularly effective for analyzing time-series patterns in power consumption [16].

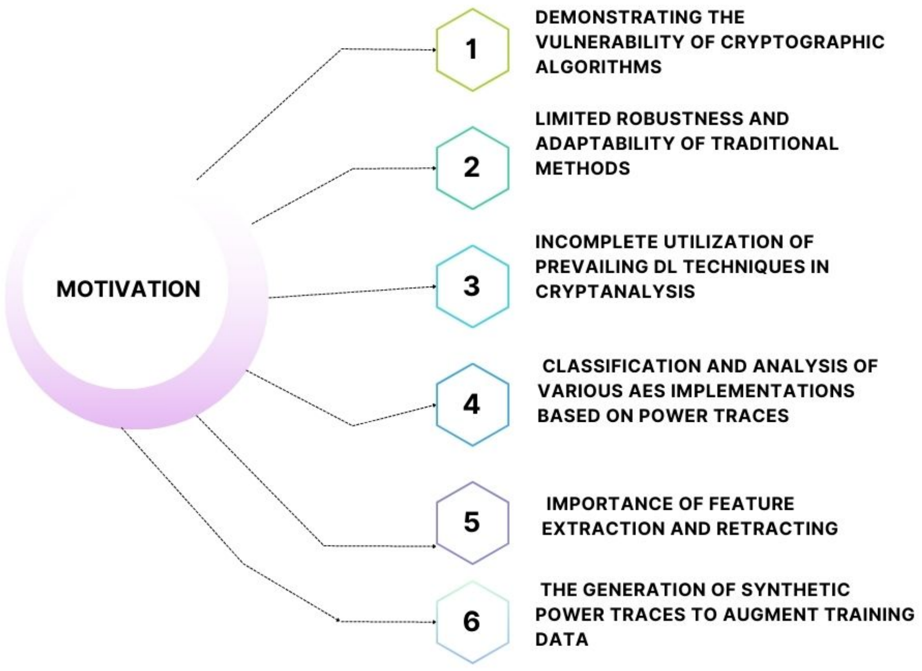

MOTIVATION
1
DEMONSTRATING THE VULNERABILITY OF CRYPTOGRAPHIC ALGORITHMS
2
LIMITED ROBUSTNESS AND ADAPTABILITY OF TRADITIONAL METHODS
3
INCOMPLETE UTILIZATION OF PREVAILING DL TECHNIQUES IN CRYPTANALYSIS
4
CLASSIFICATION AND ANALYSIS OF VARIOUS AES IMPLEMENTATIONS BASED ON POWER TRACES
5
IMPORTANCE OF FEATURE EXTRACTION AND RETRACTING
6
THE GENERATION OF SYNTHETIC POWER TRACES TO AUGMENT TRAINING DATA

Figure 1.2: Six Key Motivating Factors Driving Proposed Research Work

1.1 Motivation

The landscape of cryptography is rapidly evolving, making it a crucial area of interest. With the increasing reliance on digital systems to store, receive, and transmit vast amounts of confidential information, it is essential to prioritize the robustness, efficiency, and adaptability of cryptographic algorithms and methods.

Fig 1.2 lists the Six key motivation factors driving the proposed research work.

1.1.1 Vulnerabilities in cryptographic algorithms

Demonstrating the vulnerability of cryptographic algorithms against SCAs is crucial. By meticulously examining power traces and using deep learning (DL) models, we can understand how attacks identify normal patterns and deduce critical information, such as cryptographic keys. Understanding the susceptibility of each AES implementation to SCAs allows us to devise effective countermeasures.

1.1.2 Challenges in traditional methods

Traditional methods like Differential Power Analysis (DPA) and Correlation Power Analysis (CPA) are rudimentary approaches for detecting and preventing SCAs. However, they face significant challenges in real-world applications, such as high sensitivity to noise and jitter. Power traces collected from equipment like oscilloscopes often contain noisy emissions, complicating analysis with DPA or CPA.

Additionally, these methods struggle to evaluate complex relationships between power measurements and cryptographic operations, as they typically rely on linear relationships and simple statistical techniques. This limitation restricts their adaptability, especially in the cryptographic domain. Moreover, traditional methods require expert domain knowledge to identify points of interest in the traces, making them time-consuming and inefficient. They also lack the robustness to handle variations in hardware or environmental conditions and may not adapt well to countermeasures implemented by system designers.

1.1.3 Underutilization of advanced DL techniques

Deep Learning (DL) has expanded into various domains, providing powerful tools for data analysis and pattern identification. Although it shows excellent potential, DL remains underutilized in cryptography. Models like CNNs and RNNs significant advantages over traditional techniques. They can automatically perform precise feature extraction from raw data, recognize complex patterns, and provide clear analysis by eliminating noise and jitters.

There are numerous unexplored DL models that can be effectively utilized. Identifying and tuning hyperparameters such as epochs, batch size, optimizer, and learning rate can help develop robust models capable of extracting sensitive information.

The strength of DL can be leveraged to develop highly efficient models that can detect and mitigate SCAs. This, in turn, can help us identify existing vulnerabilities and provide solutions to combat them.

1.1.4 Analyzing and classifying AES implementations from power traces

The Advanced Encryption Standard (AES) is one of the most commonly used symmetric key cryptographic algorithms and functions as a block cipher. The widespread usage of AES makes it an appealing target for side-channel attacks. There are different implementations of AES, namely unprotected and protected AES, each exhibiting varying levels of vulnerability to side-channel attacks. Classifying and analyzing these implementations will provide us with insights into how an attacker targets each type based on its weaknesses. This understanding is crucial for developing tailored mitigation strategies to address the specific vulnerabilities of each AES implementation.

1.1.5 Feature Extraction and Retracting to original power traces

Feature extraction is crucial in analyzing power traces, which are time-series data lacking embedded features or labels. Employing feature extraction techniques helps deduce statistical properties and identify relationships relevant to AES classification and key deduction. Traditional methods require manual feature selection, which is cumbersome and relies heavily on domain expertise. In contrast,DL models streamline this process by automating feature selection, eliminating the need for prior knowledge about the underlying AES implementation.

Moreover, retracing the original power traces from the derived feature set confirms the accuracy of the feature set in representing the original traces and effectively defining the cryptographic operations involved.

1.1.6 Synthetic Data to Augment Training Data

Synthetic power traces can be generated using a Python script in a controlled and repeatable environment, employing the Hamming Weight (HW) concept [17]. These synthetic traces can be compared with collected raw power traces during AES implementation. Correlating and aligning the collected and generated power traces provide insights into the accuracy and precision of the synthetic traces, aiding in their validation. This approach is particularly useful when obtaining real power traces is challenging due to a lack of domain knowledge.

Various techniques, such as Fast-Fourier Transform (FFT), Principal Component Analysis (PCA), and Random Forest, can validate the reliability of the generated traces. If the aligned traces accurately represent the real raw traces, they can augment the training data, especially when the existing dataset is insufficient for thorough analysis. This augmentation enhances the generalizability and resilience of the models in detecting and countering SCAs.

1.1.7 Lack of Mitigation Strategies

Although there has been significant progress in uncovering the vulnerabilities in various cryptographic algorithms and demonstrating how attackers assess complex relationships, there remains a deficiency in developing robust mitigation strategies to combat existing SCAs. Most current methods focus on recognizing vulnerabilities rather than implementing prevention measures. There is a pressing need to deploy mitigation strategies that can effectively protect our cryptographic algorithms from these sophisticated attacks.

1.2 Contribution

The objective of this research is to analyze power traces from AES implementations for side-channel Attacks (SCAs) to assess vulnerabilities, classify the power traces into their corresponding AES implementations, and apply deep learning (DL) models to identify the S-BOX outputs. Subsequently, the goal is to deduce the cryptographic keys and develop effective mitigation strategies to protect these algorithms from SCAs.

The work is divided into four key modules, each focusing on and exploring the potential problems as shown in Fig 1.3

- **Module 1:** Generation and Analysis of Power Trace
- **Module 2:** Classification of power traces
- **Module 3:** Key Deduction using Deep Learning (DL) Models
- **Module 4:** Mitigation Strategies against SCAs

1.2.1 Module 1: Generation and analysis of power traces

The first module of the research focuses on the generation and detailed analysis of real and synthetic power traces collected while running a masked AES implementation. The real power traces are collected using an emulator setup that involves running the masked AES implementation on the ARDUINO platform using an ARDUINO UNO, and then collecting those power traces with the Syscomp device, which serves as a digital oscilloscope. To mimic the real power traces, synthetic power traces are generated using a Python script that employs the Hamming Weight concept to create a dataset with traces, ciphertext, plaintext, S-Box output, and a 128-bit key.

We implement a wide variety of techniques such as Fast Fourier Transform (FFT), Principal Component Analysis (PCA), linear regression, and Random Forest to align the real power traces with the synthetically generated traces and to smoothen the generated power traces. This step validates whether the synthetically generated power traces can be utilized as an alternative to real traces for augmenting the training data,

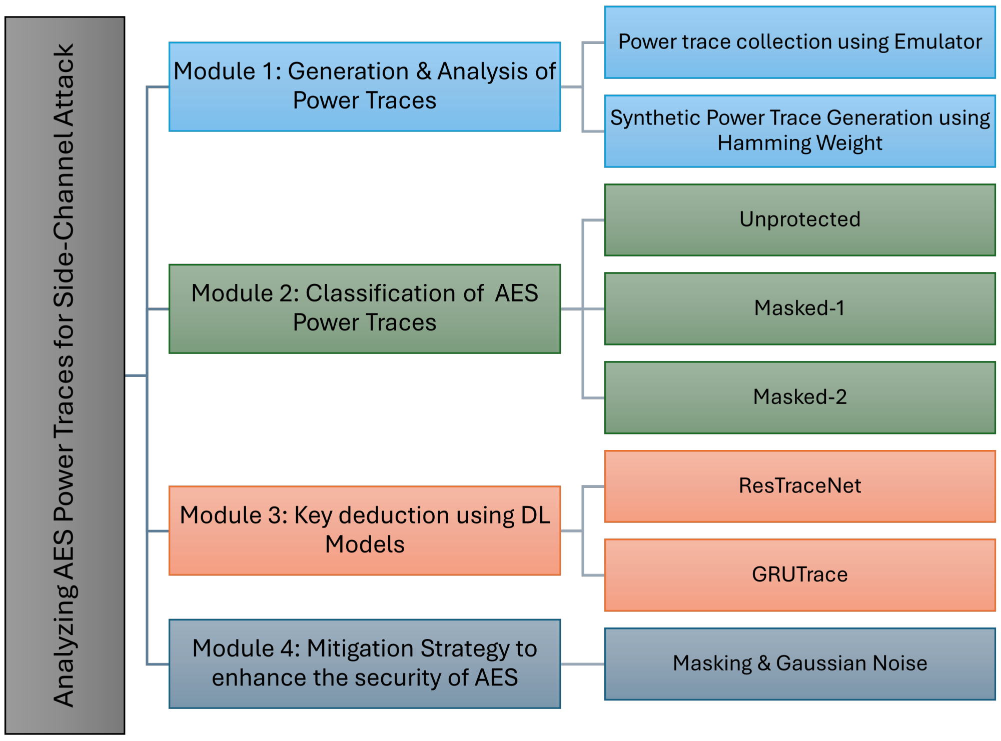

Analyzing AES Power Traces for Side-Channel Attack
Module 1: Generation & Analysis of Power Traces
Power trace collection using Emulator
Synthetic Power Trace Generation using Hamming Weight
Module 2: Classification of AES Power Traces
Unprotected
Masked-1
Masked-2
Module 3: Key deduction using DL Models
ResTraceNet
GRUTrace
Module 4: Mitigation Strategy to enhance the security of AES
Masking & Gaussian Noise

Figure 1.3: The Four Key Modules in Analysing AES Power Traces for SCA

provided they exhibit the behavior of the real traces collected.

1.2.2 Module 2: Feature extraction, classification and retracing of power traces

The second module explores the classification of power traces into various AES implementations, namely Unprotected, Masked1, and Masked2 (both of which fall under the category of protected). The classification is carried out after applying multiple feature extraction methods such as Mutual Information, Recursive Feature Elimination (RFE), Permutation Importance (PI), PCA, and more. The extracted statistical properties—mean, standard deviation, minimum, and maximum—are then fed as features into the Support Vector Machine (SVM) classifier to classify the power traces. An optimal threshold that differentiates each class is also calculated based on the probability score.

To ensure the integrity and reliability of the classifier and the classification process, we apply hashing and metadata techniques to retract the original power trace from the derived feature set containing the statistical properties.

1.2.3 Module 3: Key deduction using ResTraceNet and GRUTrace

The third module provides an in-depth exploration of key deductions using DL models. We employ advanced DL models such as ResTraceNet and GRUTrace on the ASCAD Dataset. These models are selected based on their ability to work with sequential data, uncovering complex and non-linear latent relationships, which makes them ideal candidates for key prediction.

These models are then tested on the attack dataset to correctly predict the S-box output and subsequently deduce the key. To further test the capability and reliability of our models, we select 100 power traces and test both our ResTraceNet and GRUTrace models to determine if they can accurately extract the key.

1.2.4 Module 4: Mitigation strategies

The last module addresses the problem of mitigation strategies to protect the AES algorithm against SCAs. We propose three different approaches:

- Adding Gaussian Noise to the power traces
- Adding Gaussian Noise and then masking the Ciphertext(CT)
- Adding Gaussian Noise and Structured Masking (Masking Plaintext, encrypting and masking CT)

These methods are deployed to ensure that the relationship between the power traces and cryptographic operations remains as obscure and complex as possible, making it difficult for attackers to deduce information and identify the hidden relationships between them.

1.3 Organization of the Thesis

Chapter 2: Background and Related Studies: In this chapter, we review the required definitions and primitives of side-channel attacks (SCAs), including simple power analysis (SPA), differential power analysis (DPA), and correlation power analysis (CPA). Additionally, we explain the AES algorithm and its susceptibility to SCAs. We delve into the advantages of deep learning over traditional methods in cryptanalysis. We also present research on deep learning-based side-channel analysis. We conclude the chapter by introducing the ASCAD dataset and its relevance to our study.

Chapter 3: Generation and Analysis of Power Traces: This chapter focuses on the methodology of generating and analyzing power traces. First, we discuss the collection of real power traces using the Syscomp CGR-101 oscilloscope and the generation of synthetic traces using the Hamming weight model. We then detail the timing analysis techniques used to align and correlate these traces, and we evaluate the effectiveness of synthetic data in augmenting real data. Finally, we present the results of our analysis, including the high correlation achieved using Random Forest regression.

Chapter 4: Feature Extraction, Classification, and Retracing of Power Traces: In this chapter, we describe the feature extraction techniques applied to improve the classification accuracy of AES implementations. We explain methods such as permutation importance, recursive feature elimination, L1 regularization, mutual information, and SHAP. We detail the use of Principal Component Analysis (PCA) to select the optimal features and the application of Support Vector Machine (SVM) for classification. Finally, we discuss the retracing of original power traces using hashing and the ID method, and we present the results of our classification experiments.

Chapter 5: Key Deduction using ResTraceNet and GRUTrace Models: This chapter explores the use of deep learning models, specifically ResTraceNet and GRUTrace, for predicting the S-box output and deducing the cryptographic key. We describe the datasets used, including the problems encountered with the AES_PT dataset and the selection of the ASCAD dataset. We provide a detailed explanation of the ResTraceNet and GRUTrace architectures, the training process, and the testing on attack phase data. We present the results of key deduction, including the successful extraction of the third key byte using 100 power traces. We conclude with a comparative analysis of our results with existing work.

Chapter 6: Mitigation Strategies: In this chapter, we examine and evaluate various mitigation strategies to protect AES implementations from SCAs. We introduce three methods: adding Gaussian noise to the power traces, masking ciphertext and adding Gaussian noise, and structured masking with controlled noise addition. We detail the methodology and implementation of each strategy and present the results of their effectiveness. We conclude with a comparative analysis of our proposed mitigation strategies against existing methods.

Chapter 7: Conclusion: This chapter summarizes the research, highlighting the key contributions and findings. We discuss the limitations encountered during the study and propose directions for future work. We emphasize the significance of integrating deep learning models and mitigation strategies to enhance the security of cryptographic implementations against side-channel attacks.

Chapter 8: Limitations and future work: The final discusses limitations and future work.

Chapter 2

Background Knowledge and Literature Review

Side-Channel Attacks (SCAs) exploit unintentional information leakage from cryptographic systems, such as power usage, electromagnetic signals, and timing, to extract sensitive data. SCAs can be categorized by the type of leakage (e.g., power, electromagnetic, acoustic, thermal) and the method used (e.g., SPA, DPA, CPA, template attacks, machine learning-based attacks). Electromagnetic attacks analyze emissions to understand cryptographic operations, while power analysis attacks monitor power consumption—SPA directly observes power patterns, and DPA uses statistical methods to find key correlations. Profiling attacks involve creating a device model for targeted attacks, whereas non-profiling attacks analyze data without a prior model. Algorithms like AES, RSA, and DES are particularly vulnerable due to their distinctive power consumption patterns and emissions. Traditional methods such as DPA, CPA, and SPA face challenges like sensitivity to noise, the need for expert feature selection, and issues with scalability. Deep Learning (DL) models, including RNN, ResNet and CNN address these challenges by automating feature extraction, handling complex patterns, being robust against noise, and generalizing well across different scenarios, thus improving the detection and mitigation of SCAs.

2.1 Background Knowledge

2.1.1 Side-channel attacks

Side-channel attacks (SCAs) are a class of attacks that exploit information leaked from the physical implementation of a system rather than from software vulnerabilities or mathematical characteristics. These attacks utilize indirectly leaked information such as power emissions, electromagnetic emissions, acoustic signals, thermal energy, timing signals, and more to deduce sensitive information like cryptographic keys, outputs of various cryptographic operations, and plaintext [1].

2.1.2 History and implications

SCAs were first introduced by Paul Kocher in 1996 [18] with the concept of timing attacks, which calculated the timing variations in performing different operations and how they correlated with the input text and secret data. Before the advent of side-channel attacks by Kocher, the primary focus was on protecting systems from unauthorized access and ensuring robustness. However, Kocher's valuable findings caused a significant shift in the cryptographic community towards considering both the physical and environmental vulnerabilities of cryptographic implementations. Over time, these attacks expanded to include critical methods such as Power Analysis (PA) attacks, which have proven to be extremely powerful in breaking crucial algorithms. The non-invasive nature of these attacks, which enables an attacker to gather information without damaging or altering the system, makes them very hard to detect and stealthy. The cost-effectiveness and high success rate of SCAs, leveraging commonly available and inexpensive tools like oscilloscopes and probes, make them a preferred choice for attackers [2].

The implications of SCAs are immense, as they undermine the security, confidentiality, and integrity of cryptographic systems without needing to compromise the underlying algorithm. This approach allows attackers to bypass traditional security measures, which mainly focus on algorithmic robustness rather than targeting the physical properties of the system [9]. SCAs follow a structured approach that begins with data collection, where side-channel information such as electromagnetic emissions, power traces, or thermal traces are gathered from the targeted device. The second phase involves analyzing the collected data using statistical techniques or machine learning methods to identify correlations or hidden patterns that can reveal sensitive information. The final step is deducing the critical information based on the analysis and insights generated, thereby compromising the security goals of the target device [19].

2.1.3 Types of SCAs

SCAs are classified based on the type of information they leak and the attack methodology used to carry out the attack, which is critical for understanding underlying vulnerabilities and developing mitigation strategies. Channel information leakages include several types, such as power analysis, electromagnetic analysis, acoustic analysis, timing analysis, and thermal analysis. Power analysis monitors the power consumption patterns of the target device to extract sensitive information and includes subdivisions such as Simple Power Analysis (SPA), which observes power consumption to glean cryptographic operations, and Differential Power Analysis (DPA), which employs statistical methods to find correlations between power consumption and processed data [20]. Electromagnetic analysis collects the electromagnetic emissions generated by electronic components during cryptographic operations to deduce internal state details, revealing critical information like cryptographic keys [21]. Acoustic analysis uses the sound produced by a device's electronic or mechanical components during operation, collecting these acoustic signals with high-sensitivity microphones to identify subtle differences in sound waves that expose operational states and specific instructions being executed [22]. Timing attacks exploit the time it takes to execute cryptographic operations, relying on the fact that different operations take varying amounts of time based on input data and the internal state of the system [23]. Finally, thermal analysis involves assessing the heat emitted during cryptographic operations and how it varies with different computational processes to infer the operations being performed. Thermal attacks are the least commonly used approach. [24]

Based on the methodology used for the attack, SCAs are classified into profiling and non-profiling attacks. Profiling attacks consist of two phases: the profiling phase and the attack phase. In the profiling phase, the attacker creates a device similar to the target device and conducts experiments under varying conditions to understand the system's behavior, including normal side-channel information. This information is leveraged to build a detailed model that characterizes how the side-channel information pertains to the internal state of the device, including various operations or keys. Once the profiling phase is complete, the model is used to analyze the attack data collected from the actual target device to deduce critical information [25]. Examples of profiling attacks include template attacks (TA), machine learning (ML)-based attacks, and deep learning (DL)-based attacks. In a template attack, a template of the target system is created during the profiling phase, capturing the critical relationship between the obtained side-channel information and cryptographic operations. During the attack phase, this template is applied to the actual data to deduce information with high accuracy. DL-based attacks use DL models such as CNNs, RNNs and GRU to extract essential features and learn complex relationships, making them applicable to real-world scenarios [26].

In contrast, non-profiling attacks do not have a profiling phase for model building. Instead, they directly analyze the information collected from the target system to glean sensitive information. During the data collection phase, the attacker gathers side-channel information and then applies data analysis techniques, ranging from Estimation of distribution algorithms (EDA) to statistical and analytical methods, to identify and extract relevant patterns that reveal critical information [27]. Since they do not use a predefined model, non-profiling attacks rely on methods such as differential power analysis (DPA) and correlation power analysis (CPA). DPA involves gathering multiple power traces while the system performs cryptographic operations with varying inputs. CPA is an enhanced version of DPA that uses correlation coefficients to compute linear relationships, offering greater accuracy in identifying key-dependent power consumption patterns [28]

2.1.4 Profiling vs non-profiling attacks

Both profiling and non-profiling attacks vary greatly concerning accuracy, complexity, resource requirements, and adaptability. Profiling attacks achieve greater accuracy due to the presence of the profiling phase that requires extensive knowledge of the cryptographic system but is relatively more complex and takes longer duration to carry out [29]. In contrast, non-profiling attacks are simpler and easier to set up since they do not have a profile-building phase but may tend to be less accurate and precise. Profiling attacks require a larger amount of resources, including computational time, power, and access to a device that is similar to the target device. On the other hand, non-profiling attacks require fewer resources in terms of the pre-attack setup phase but would require vast quantities of power traces to secure accurate results. Moreover, profiling attacks are extremely adaptable to various characteristics of the system which makes them more reliable. Non-profiling attacks might struggle with robustness and adaptability as they might not be able to deal with variations in the device [25]. Fig. 2.1 shows how a profiling and non-profiling attack works.

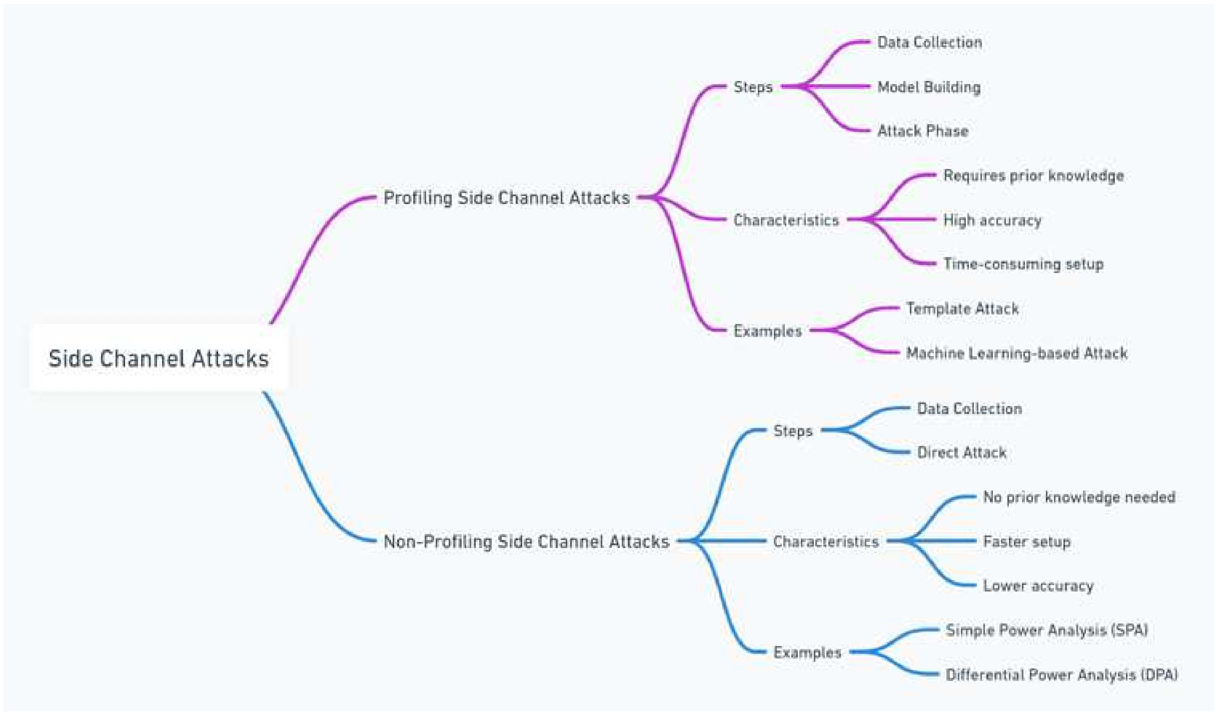


Figure 2.1: Profiling vs non-profiling attacks

2.1.5 Electromagnetic emission analysis and power analysis

We will dive deeper into electromagnetic emissions (EM) and power analysis, which form the core of this research. EM attacks target the electromagnetic emissions generated by electronic devices during their operations. These emissions are generally captured using specialized instruments such as antennas, probes, and oscilloscopes. The captured emissions are then analyzed to infer information about the internal state and functioning of the target device. The major advantage of EM attacks is that they can be conducted from a distance without requiring physical contact, making them difficult to detect [30]. On the other hand, power analysis attacks monitor the power consumption of a device while it is performing cryptographic operations. These attacks leverage the concept that the power consumption pattern of a device can change depending on the data being processed and the operation being carried out. By gathering and assessing such power traces—patterns of power consumption over time—key information can be deduced [31]. Fig. 2.2 shows a sequence diagram indicating how a power analysis attack is carried out.

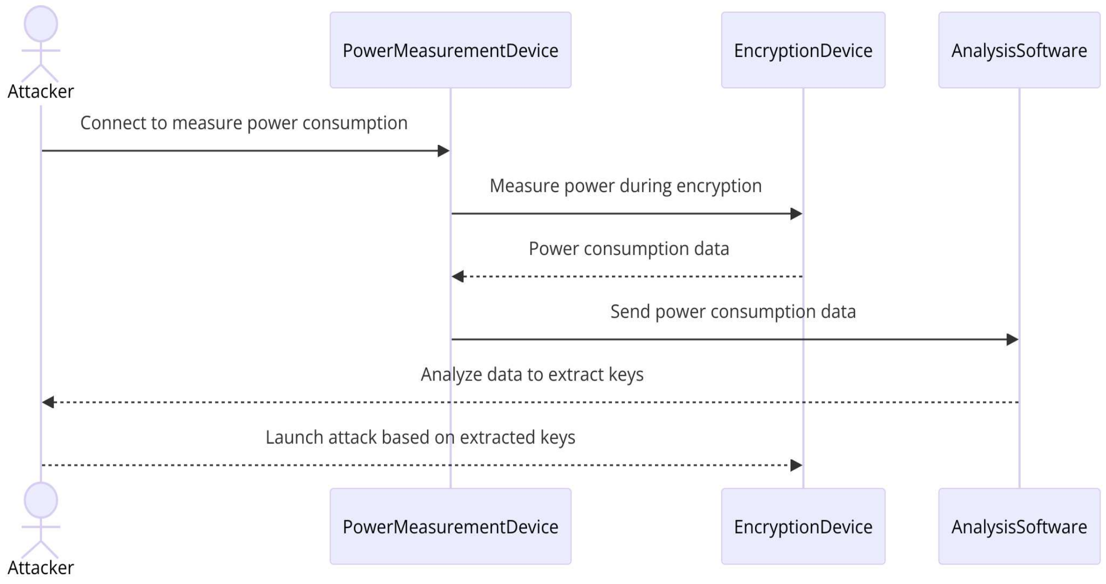

Attacker
PowerMeasurementDevice
EncryptionDevice
AnalysisSoftware
Connect to measure power consumption
Measure power during encryption
Power consumption data
Send power consumption data
Analyze data to extract keys
Launch attack based on extracted keys
Attacker
PowerMeasurementDevice
EncryptionDevice
AnalysisSoftware

Figure 2.2: Power analysis attack

2.1.6 Simple power analysis, differential power analysis and correlation power analysis

PA attacks are divided into several types, with the most popular ones being Simple Power Analysis (SPA), Differential Power Analysis (DPA), and Correlation Power Analysis (CPA). SPA directly measures the power consumption of a cryptographic system while it is performing its operations. The visual patterns of the power traces are then analyzed to find their correlation to cryptographic operations [31]. For example, operations such as S-BOX, MixColumns, and ShiftRows in an AES implementation might produce distinct signatures, and the challenge is to identify these patterns and deduce information.

DPA is a more sophisticated and resilient technique compared to SPA. In DPA, the input is varied while the device performs various cryptographic operations, and a large number of power traces are collected. The main concept is to statistically identify variations in power traces as the inputs change. The observed power consumption can be correlated with hypothetical models built to identify key bits [32].

CPA builds upon the capabilities of DPA by applying more refined and enhanced statistical techniques, such as the Pearson Correlation Coefficient, which calculates the linear relationship between power consumption measurements and hypothetical power consumption models. A power consumption model is created based on Hamming weight (HW) or Hamming distance (HD) using hypothetical values of the cryptographic key. The correlation between this generated model and real traces is then analyzed, with the key hypothesis that yields the highest correlation likely being the correct key. CPA is generally more powerful than DPA as it can identify even the smallest correlations [33].

2.1.7 DPA vs CPA

Both DPA and CPA are quite similar in their approach, but they differ in parameters such as methodologies, accuracy, and robustness. DPA can be less precise and more susceptible to noise and jitter, making it less effective in environments with significant external interference. In contrast, CPA uses a correlation-based approach to accurately identify key bytes, which makes it robust against external interference. CPA can handle higher-dimensional data better than DPA, resulting in a higher degree of accuracy. However, a major challenge of using CPA, which also applies to DPA, is the vast quantity of data and computational resources required to perform the correlation analysis [34]

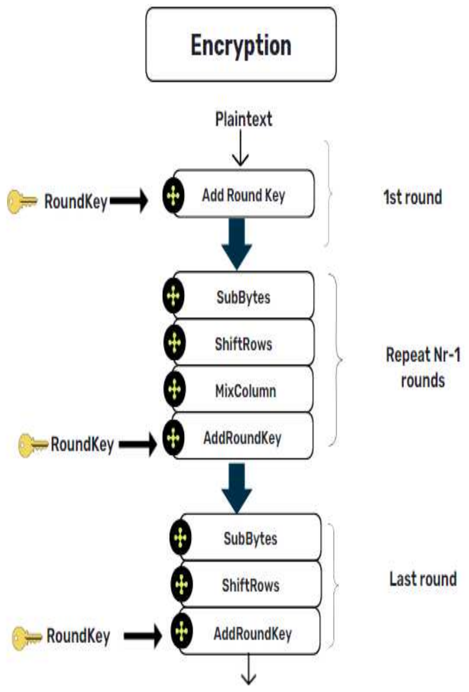
2.1.8 Cryptographic algorithms vulnerable to SCAs

Cryptographic algorithms, classified into asymmetric and symmetric key algorithms, can both be compromised by SCAs. Examples of symmetric algorithms include the Advanced Encryption Standard (AES), the Data Encryption Standard (DES), and Triple DES (3DES). Prominent examples of asymmetric algorithms include Rivest-Shamir-Adleman (RSA), Elliptic Curve Cryptography (ECC), and Diffie-Hellman (DH). AES is particularly renowned for its extensive use in securing data, yet it is susceptible to timing and power analysis attacks. DES remains vulnerable to DPA and is rarely used today due to its inherent weaknesses. 3DES is also vulnerable to power analysis techniques like DPA and CPA, which can extract keys based on the visualization and analysis of power traces. RSA is exposed to timing attacks because of the various mathematical operations involved, while ECC is vulnerable to power analysis and electromagnetic emission attacks. DH is predominantly used for key exchange and is mostly exposed to timing attacks [35].

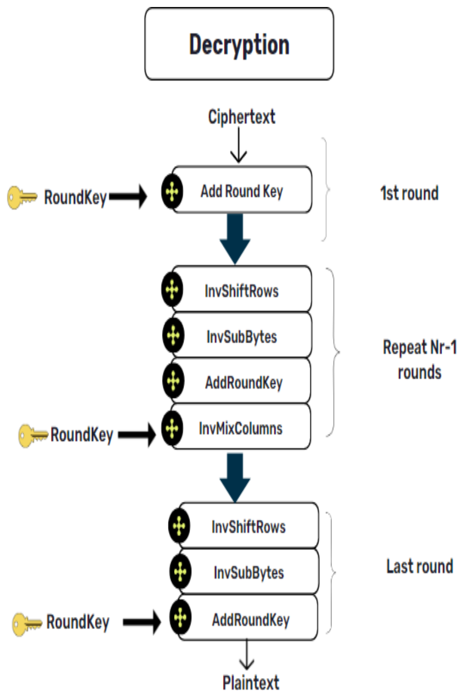
2.1.9 AES algorithm

AES is a symmetric key cryptographic algorithm widely known for its robustness, efficiency, and flexibility. Adopted as a major standard by the National Institute of Standards and Technology (NIST) in 2001, AES is commonly used in a broad range of applications, from securing online communications to encrypting sensitive information. As a block cipher, AES operates on fixed-size blocks and supports key sizes of 128, 192, and 256 bits [36].

AES operates on a 4x4 matrix, referred to as the state, and involves both encryption and decryption processes, with the number of rounds depending on the key size. The process begins with key expansion, where the initial key is expanded to produce unique keys for every round of encryption. The state matrix is then combined with the first round key using a bitwise XOR operation, establishing a relationship between the input data and the encryption key. Each round comprises four phases: SubBytes, ShiftRows, MixColumns, and AddRoundKey. In the SubBytes phase, each byte is substituted with its corresponding byte from an S-box to introduce non-linearity. During ShiftRows, each row of the matrix is cyclically shifted to the left. In the MixColumns phase, columns are treated as polynomials and multiplied by a fixed polynomial to provide diffusion. Lastly, the AddRoundKey phase combines the state matrix with a round key using a bitwise XOR operation, adding confusion. In the final round, all operations are performed except for MixColumns. The decryption process involves reversing these steps using the same round keys but applying inverse transformations to reproduce the plaintext [37]. Fig 2.3 shows the encryption and decryption operation of an unprotected AES implementation. 1 shows the pseudocode for an unprotected AES encryption implementation.



(a) Encryption operation of AES Implementation



(b) Decryption operation of AES Implementation

Figure 2.3: Encryption and decryption operation in AES implementation

2.1.10 AES against SCAs

Although AES is a powerful algorithm, it is particularly vulnerable to SCAs due to the power consumption patterns and EM emissions generated during each cryptographic operation. The basic structure of AES, with its multiple rounds of substitution, permutation, and key scheduling, causes different kinds of data leakage, as the power consumption changes based on the data and operations being carried out. These leakages can be exploited by attackers, making AES a prime target for SCAs [38].

Algorithm 1 AES Encryption Algorithm pseudocode

```
Input: plaintext, key
Output: ciphertext
1 Function AESencrypt(plaintext, key):
2     blocks := divideIntoBlocks(plaintext) roundKeys := getRoundKeys(key)
      foreach block in blocks do
3         addRoundKey(block, roundKeys[0]) for $i \leftarrow 1$ to $(numRounds - 1)$ do
4             subBytes(block) shiftRows(block) mixColumns(block)
               addRoundKey(block, roundKeys[i])
5         end
6         subBytes(block) shiftRows(block) addRoundKey(block,
           roundKeys[numRounds])
7     end
8     ciphertext := reassemble(blocks) return ciphertext
```

EM attacks capture the power emissions emitted during the execution of the AES algorithm on a target device using electronic equipment such as oscilloscopes or picoscopes. These emissions are correlated with the internal state of the AES algorithm and can be used to deduce cryptographic keys, S-box outputs, or plaintext. The emissions can be captured without being in close proximity to the target device, making it a very difficult attack to detect and counter. This, in turn, makes EM attacks an extremely powerful tool for breaking AES implementations, especially when combined with complex side-channel analysis techniques such as DPA, CPA, and deep learning-based techniques [39].

2.1.11 Drawbacks of traditional methods

While DPA, CPA, and SPA are valuable tools for conducting side-channel attacks, they have significant limitations that can affect their real-world applicability. SPA relies on the visual analysis of collected power traces to deduce information, meaning it heavily depends on the quality of the power traces, which must be clear and distinct—an expectation that might not be feasible in all cases. Moreover, SPA does not use statistical measures to identify patterns, resulting in less detailed information compared to DPA and CPA [40].

DPA, one of the more sophisticated techniques, faces challenges due to its sensitivity to noise, limiting its applicability in real-world scenarios. In practice, power traces might be generated in contaminated environments, degrading the quality of the collected data and exposing it to noise and other operations, making it difficult to perform DPA. Additionally, DPA requires a large amount of data to perform reliable analysis, making it resource-constrained and time-consuming. It also assumes a linear relationship between the power traces and the internal state of the algorithm, which might not always be true and could fail to capture complex relationships. Similarly, CPA, while more refined than DPA, is also susceptible to noise, which can obscure unseen patterns. It requires heavy computational resources due to the complex statistical operations involved, making it challenging for resource-constrained devices. CPA's heavy reliance on the hypothetical power model can also reduce accuracy; any inaccuracies or deficiencies in the model can pose significant problems [41]. Common limitations across all these methods include the need for domain experts to interpret the collected power traces and identify points of interest, making them less accessible to non-specialists. They also struggle with high-dimensional data and large volumes of data, which can compromise performance and efficiency, leading to scalability issues. Lastly, these methods are not effective against modern devices with strong countermeasures such as random delays, masking, and other forms of obfuscation.

2.1.12 Limitations of Machine Learning(ML)

Machine learning (ML) techniques have been extensively used for various tasks in pattern recognition and data analysis. However, they possess certain limitations when applied to complex tasks like side-channel attacks (SCA) analysis on AES implementations. Recognizing these limitations is crucial for justifying the shift towards more sophisticated models like Deep Learning (DL).

Traditional ML models often rely heavily on handcrafted features or require domain expertise to design effective features for the specific task. This approach can limit the model's ability to capture complex patterns inherent in power trace data, which are crucial for detecting subtle cryptographic operations [42]. In contrast, DL models, particularly CNNs and RNNs, excel in automatic feature extraction. They can learn to identify intricate patterns directly from raw data, thus eliminating the need for manual feature engineering.

ML models can struggle with scalability and flexibility when dealing with high-dimensional data or when the data distribution changes over time. This is particularly problematic in side-channel analysis where electronic noise and device variations can alter data characteristics. DL models are inherently more scalable and can adapt to new, unseen variations in data through transfer learning and fine-tuning techniques. This makes them more robust to changes in the dataset and device characteristics [43].

Most ML models do not natively handle sequential and temporal dependencies in data, which are critical in analyzing time-series data like power traces. DL models like LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Units) are designed to process sequences with complex dependencies over time. This capability is vital for effective modeling of power traces where temporal patterns directly relate to cryptographic processes [44].

ML models often have limited generalization capabilities, particularly when trained on datasets that do not represent the full spectrum of real-world scenarios. This can lead to poor performance when the model is applied outside the scope of its training data. DL models generally provide better generalization due to their deeper and more complex architectures. They are capable of learning more abstract representations that are effective across a broader range of conditions [45].

While ML models are typically easier to train and require less computational resources than DL models, they often require more expertise in selecting and tuning the hyperparameters to achieve optimal performance. Although DL models are computationally intensive, recent advances in hardware accelerations (like GPUs and TPUs) and software frameworks have significantly reduced these challenges, making it feasible to train complex models efficiently.

Given these considerations, our decision to employ DL models over ML models is driven by the need to handle high-dimensional, sequential data with higher accuracy and generalization. DL models' ability to automatically extract meaningful features from raw data and their robustness to variations in data make them particularly suited for detecting and mitigating side-channel attacks on cryptographic implementations. This approach aligns with the cutting-edge of research in the field, where DL continues to push the boundaries of what's possible in security analytics.

2.1.13 Advantages of Deep Learning (DL) over traditional methods

Deep Learning (DL) models have advanced capabilities that can address the challenges posed by DPA, SPA, and CPA. Traditional methods heavily rely on manual feature extraction, where domain experts identify points of interest and relevant features from the power traces. This process is time-consuming and heavily dependent on domain knowledge. In contrast, DL models can automatically identify and extract essential features from raw power traces. Models such as CNN and RNN can learn hierarchical feature representations from the provided input without explicit human interference, reducing the dependency on domain experts and streamlining the analysis [46].

Traditional SCA techniques are based on linear relationships between power consumption and cryptographic operations, limiting their ability to capture complex non-linear relationships that are crucial. DL models excel at utilizing non-linear relationships due to their robust architectures and use of activation functions. Models such as CNN and RNN can capture both spatial and temporal dependencies, thereby enhancing the accuracy and reliability of SCAs. A major challenge for power analysis techniques is noise and variations inherent in the power traces. DL models are robust to noise because they can learn and generalize from large datasets. By training on a diverse set of noisy and clear raw traces, DL models can filter out unwanted noise and jitter, revealing the underlying latent patterns. Data augmentation techniques can also be used to generate synthetic power traces, increasing the diversity and size of the training set, which further boosts the models' robustness against external interference [47].

One of the greatest advantages of DL models is their ability to generalize from large datasets. Once trained on a substantial volume of data, these models can adapt to other variations of cryptographic algorithms and attack scenarios without requiring major modifications, providing a versatile solution across various circumstances. DL models can handle high-dimensional data and large volumes of power traces, unlike traditional methods. DPA and CPA tend to struggle with scalability issues because they rely heavily on statistical properties and techniques. In contrast, DL models can utilize advanced optimization techniques and hardware acceleration, such as GPUs, to process large datasets efficiently [48].

Finally, many modern cryptographic devices implement strong countermeasures like masking, delays, and noise injection to combat SCAs. Traditional methods find it challenging to bypass such security mechanisms, while DL models, with their ability to identify new complex patterns and their continuous learning capabilities, are well-equipped to overcome these barriers.

2.1.14 DL models

Among the many deep learning models, ResNet and GRU have gained notable popularity for their ability to handle complex patterns. ResNet, which is based on convolutional neural network (CNN) architecture, excels at processing spatial data due to its deep layer structure. In contrast, GRU, a type of recurrent neural network (RNN), is particularly effective at sequential data modeling, capturing time-dependent nuances in datasets [49].

ResNet Architecture

ResNet is a type of deep neural network (DNN) designed for training deep architectures and addresses the problem of vanishing gradients, which can hinder the training of deep networks. It consists of residual blocks that ensure the network learns residual functions with respect to input layers rather than unreferenced functions. ResNets have demonstrated exceptional performance in image recognition tasks and can learn intricate hierarchical feature representations, enhancing accuracy in key prediction [50].

A ResNet (Residual Network) architecture is a kind of deep neural network which is developed to address the vanishing gradient problem by bringing in skip connections or shortcuts that enables gradients to flow more smoothly during backpropagation

[51]. The architecture starts with an initial Conv1D layer with 64 filters, a kernel size of 7, and a stride of 2. This layer is succeeded by a batch normalization layer and a LeakyReLU activation function [52]. Batch normalization is a technique that normalizes the input of each layer, maintaining the mean output close to 0 and the output standard deviation close to 1. This helps in the training process and improving the stability of the network. The LeakyReLU activation function is a variant of the Rectified Linear Unit (ReLU) that allows a small, non-zero gradient when the unit is not active, which helps to mitigate the problem of dead neurons and improves the learning capability of the network [52]. The network then includes a series of residual blocks, each containing two Conv1D layers with filters, kernel sizes, and strides consistent with the block's design, typically 64, 128, 256, or 512 filters, kernel sizes of 3, and strides of 1 for the first layer and 2 for the second layer within the block. Each Conv1D layer in the residual block is followed by batch normalization and LeakyReLU activation [52]. The vanishing gradient problem occurs when gradients used to update the weights become very small, effectively preventing the network from learning. Residual blocks address this by using skip connections that allow the gradient to bypass certain layers, making it easier for the network to learn even with increased depth. Skip connections are added to each residual block to allow the input to bypass the block and be added to the output of the block [51]. After the series of residual blocks, a global average pooling layer is applied to reduce the dimensions of the feature maps before passing them to a fully connected (dense) layer. A dropout layer with a rate of 0.5 is added to prevent overfitting [53]. The final output layer uses a softmax activation function for classification tasks. The network is compiled with the Adam optimizer, which is used for its adaptive learning rate capabilities [54]. The loss function used is categorical cross-entropy, suitable for multi-class classification problems. To enhance training efficiency and prevent overfitting, a learning rate scheduler adjusts the learning rate dynamically during training, and early stopping is implemented to halt training when the validation loss stops improving [55]. The model is trained with a batch size of 32 and for a total of 100 epochs.

Gated recurrent unit (GRU) Architecture

Introduced by Cho et al. in [56], GRU is a type of RNN designed to handle sequential data. They are a simplified version of Long Short-Term Memory (LSTM) models, featuring fewer parameters and a gating mechanism. The ability to capture temporal dependencies and identify the sequential characteristics of power traces makes GRUs an ideal choice for deducing sensitive information. Both models can automatically extract features, model sequential relationships, provide robustness to noise and variations in data, and handle large volumes of data, making them ideal choices in the field of SCAs [57]. In a GRU architecture, each unit contains two gates: the reset gate and the update gate. The reset gate determines how much of the past information to forget, while the update gate controls how much of the past information to retain and how much of the new information to add. This allows the GRU to maintain long-term dependencies without explicitly storing a separate cell state [58]. A bidirectional GRU extends the standard GRU by processing the input data in both forward and backward directions. This bidirectional processing enables the network to capture information from both past and future states, enhancing its ability to understand the context in sequential data. Each bidirectional layer consists of two GRUs: one processing the input sequence in the forward direction and the other in the backward direction. The outputs of these two GRUs are concatenated to form the final output of the bidirectional layer [59]. When using the return_sequences parameter, the GRU layer outputs the full sequence of outputs for each input time step, rather than just the final output. This is particularly useful in tasks where the entire output sequence is needed, such as in sequence-to-sequence models for machine translation or speech recognition [58].

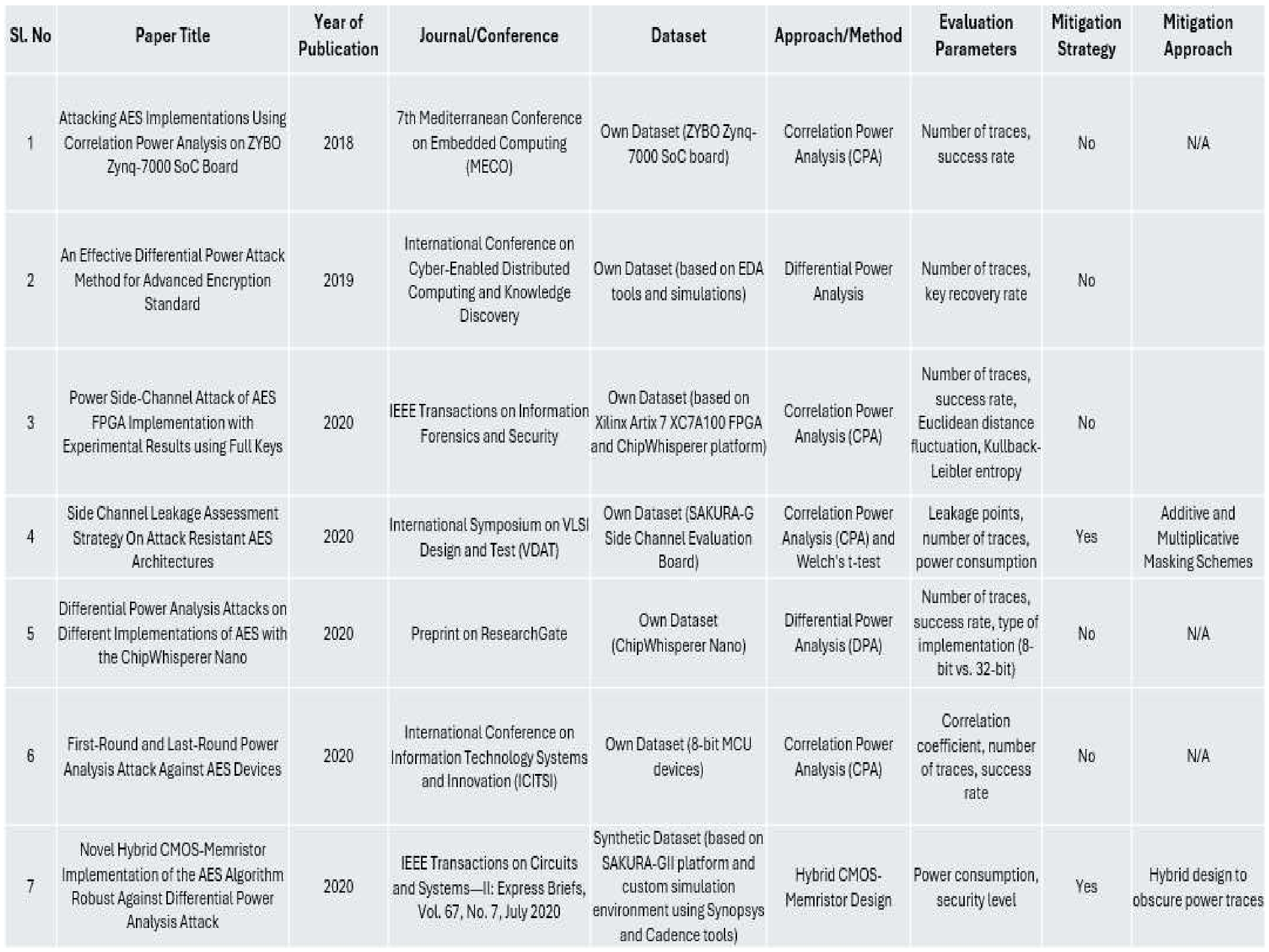
To improve the generalization and prevent overfitting, a dropout layer is applied with a typical dropout rate of 0.5 [60]. Layer normalization is also applied, which normalizes the activations of the neurons in a layer to improve training speed and performance stability [61].

The loss function used is categorical cross-entropy, suitable for multi-class classification problems [62]. To enhance training efficiency and prevent overfitting, a learning rate scheduler such as ReduceLROnPlateau is used to reduce the learning rate when the validation loss plateaus, and early stopping is implemented to halt training when the validation loss stops improving [63]. The model is trained with a typical batch size of 32 and for a total of 100 epochs [62].

2.2 Literature Review

2.2.1 Differential Power Analysis (DPA) and Correlation Power Analysis (CPA)

Fig. 2.4 illustrates various studies that have been conducted using traditional methods such as DPA and CPA for side-channel attacks on AES implementations. Each entry in the table focuses on attributes such as the dataset used, the technique applied, evaluation parameters, and whether a mitigation strategy was employed.



Sl. No	Paper Title	Year of Publication	Journal/Conference	Dataset	Approach/Method	Evaluation Parameters	Mitigation Strategy	Mitigation Approach
1	Attacking AES Implementations Using Correlation Power Analysis on ZYBO Zynq-7000 SoC Board	2018	7th Mediterranean Conference on Embedded Computing (MECO)	Own Dataset (ZYBO Zynq-7000 SoC board)	Correlation Power Analysis (CPA)	Number of traces, success rate	No	N/A
2	An Effective Differential Power Attack Method for Advanced Encryption Standard	2019	International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery	Own Dataset (based on EDA tools and simulations)	Differential Power Analysis	Number of traces, key recovery rate	No	
3	Power Side-Channel Attack of AES FPGA Implementation with Experimental Results using Full Keys	2020	IEEE Transactions on Information Forensics and Security	Own Dataset (based on Xilinx Artix 7 XC7A100 FPGA and ChipWhisperer platform)	Correlation Power Analysis (CPA)	Number of traces, success rate, Euclidean distance fluctuation, Kullback-Leibler entropy	No	
4	Side Channel Leakage Assessment Strategy On Attack Resistant AES Architectures	2020	International Symposium on VLSI Design and Test (VDAT)	Own Dataset (SAKURA-G Side Channel Evaluation Board)	Correlation Power Analysis (CPA) and Welch's t-test	Leakage points, number of traces, power consumption	Yes	Additive and Multiplicative Masking Schemes
5	Differential Power Analysis Attacks on Different Implementations of AES with the ChipWhisperer Nano	2020	Preprint on ResearchGate	Own Dataset (ChipWhisperer Nano)	Differential Power Analysis (DPA)	Number of traces, success rate, type of implementation (8-bit vs. 32-bit)	No	N/A
6	First-Round and Last-Round Power Analysis Attack Against AES Devices	2020	International Conference on Information Technology Systems and Innovation (ICITSI)	Own Dataset (8-bit MCU devices)	Correlation Power Analysis (CPA)	Correlation coefficient, number of traces, success rate	No	N/A
7	Novel Hybrid CMOS-Memristor Implementation of the AES Algorithm Robust Against Differential Power Analysis Attack	2020	IEEE Transactions on Circuits and Systems—II: Express Briefs, Vol. 67, No. 7, July 2020	Synthetic Dataset (based on SAKURA-GII platform and custom simulation environment using Synopsys and Cadence tools)	Hybrid CMOS-Memristor Design	Power consumption, security level	Yes	Hybrid design to obscure power traces

Figure 2.4: Literature review on Non-profiling methods

- **Attacking AES Implementations Using Correlation Power Analysis on ZYBO Zynq-7000 SoC Board:** This study used a dataset from the ZYBO Zynq-7000 SoC board and employed CPA to assess the number of traces and success rate. No mitigation strategy was implemented [64].

- **An Effective Differential Power Attack Method for Advanced Encryption Standard:** This research utilized a proprietary dataset created using Estimation of Distribution Algorithms (EDA) tools and simulations, applying DPA to determine the number of traces needed and the key recovery rate. No mitigation strategy was employed [65].

- **Power Side-Channel Attack of AES FPGA Implementation with Experimental Results using Full Keys:** This study used a dataset based on the Xilinx Artix 7 XC7A100 FPGA and the ChipWhisperer platform, employing CPA to analyze the number of power traces, success rate, and Euclidean distance fluctuation. No mitigation strategy was proposed [66].

- **Side-channel Leakage Assessment Strategy on Attack-Resistant AES Architectures:** This analysis used the SAKURA-G side-channel Evaluation Board, employing both CPA and Welch's t-test to identify points of interest and compute power consumption. Mitigation strategies such as additive and multiplicative masking were implemented [67].

- **Differential Power Analysis Attacks on Different Implementations of AES with the ChipWhisperer Nano:** This study used the ChipWhisperer Nano to generate the dataset, applying DPA to analyze the number of traces, success rate, and implementation type. No countermeasures were proposed [68].

- **First-Round and Last-Round Power Analysis Attack Against AES Devices:** This research used a dataset based on 8-bit MCU devices and employed DPA to evaluate the number of traces, success rate, and correlation coefficient. No mitigation strategy was mentioned [69].

- **Novel Hybrid CMOS-Memristor Implementation of the AES Algorithm Robust Against Differential Power Analysis Attack:** This study used a synthetic dataset based on the SAKURA-GII platform and custom simulation environments, employing a hybrid CMOS-Memristor design to obscure power traces. This strategy effectively mitigated power analysis attacks by enhancing security [70].

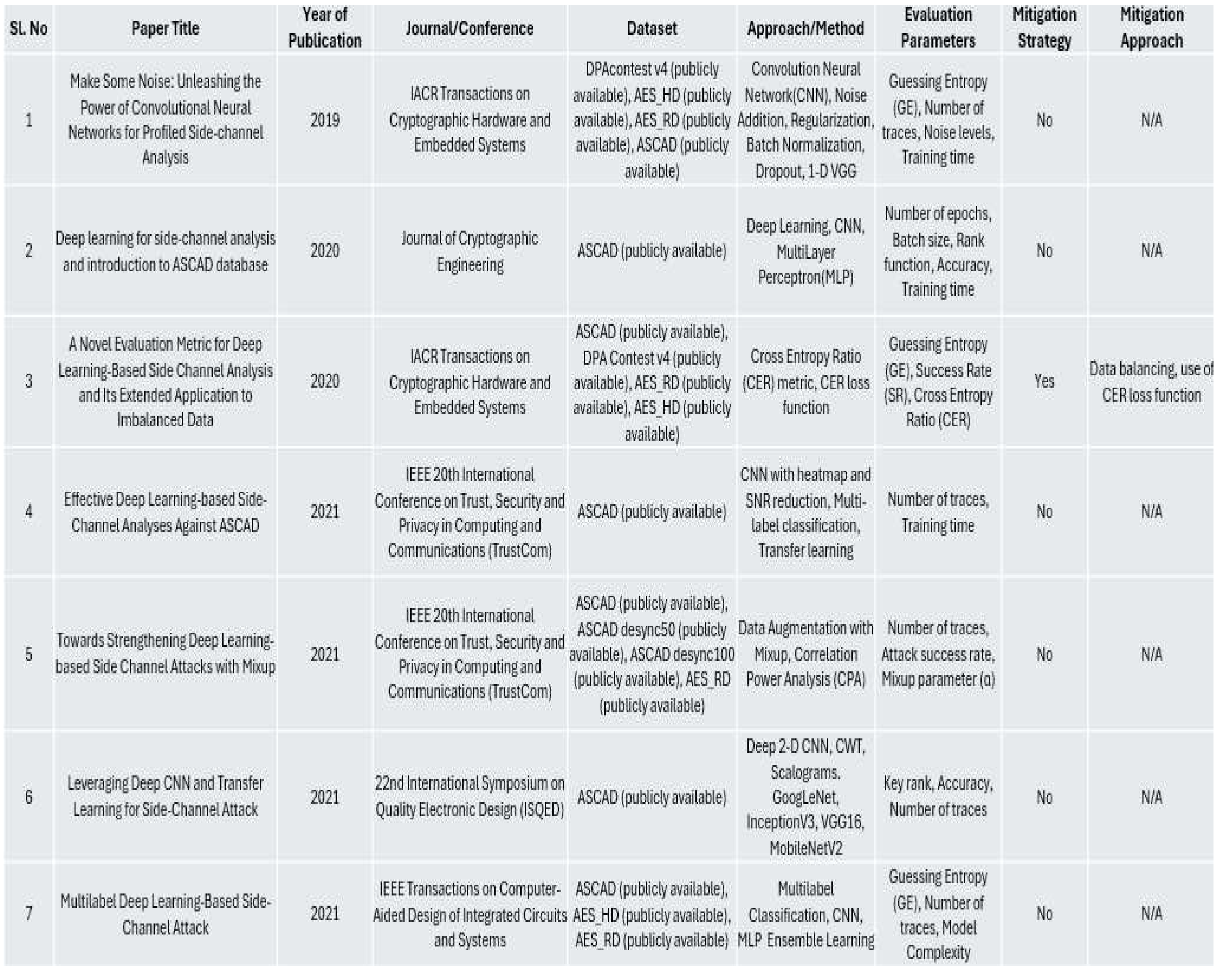
2.2.2 Deep Learning (DL) based side-channel attacks

Fig. 2.5, 2.6 and 2.7 presents a comprehensive overview of various studies employing deep learning techniques for side-channel analysis (SCA) on AES implementations. Each field explores various attributes such as the dataset used, the method leveraged, evaluation parameters, and the mitigation strategy employed.

- **Make Some Noise: Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis:** This study made use of three datasets namely DPAv4 contest dataset, AES_HD, and ASCAD datasets, utilizing models like Convolutional Neural Networks (CNN) with noise addition, regularization, batch normalization, dropout, and 1-D VGG for the assessment. The evaluation parameters consisted of guessing entropy (GE), number of traces, noise levels, and training time. No mitigation strategy was implemented [71].

- **Deep Learning for Side-Channel Analysis and Introduction to ASCAD Database:** This analysis utilized the ASCAD dataset and employed deep learning techniques such as CNN and Multi-Layer Perceptron (MLP). Evaluation parameters included the number of epochs, batch size, rank function, accuracy, and training time. No countermeasures were employed [72].

- **A Novel Evaluation Metric for Deep Learning-Based Side-Channel Analysis and Its Extended Application to Imbalanced Data:** This study leveraged ASCAD, AES_RD, and DPAv4 contest datasets, introducing a cross-entropy ratio (CER) metric and CER loss function for evaluation. Mitigation



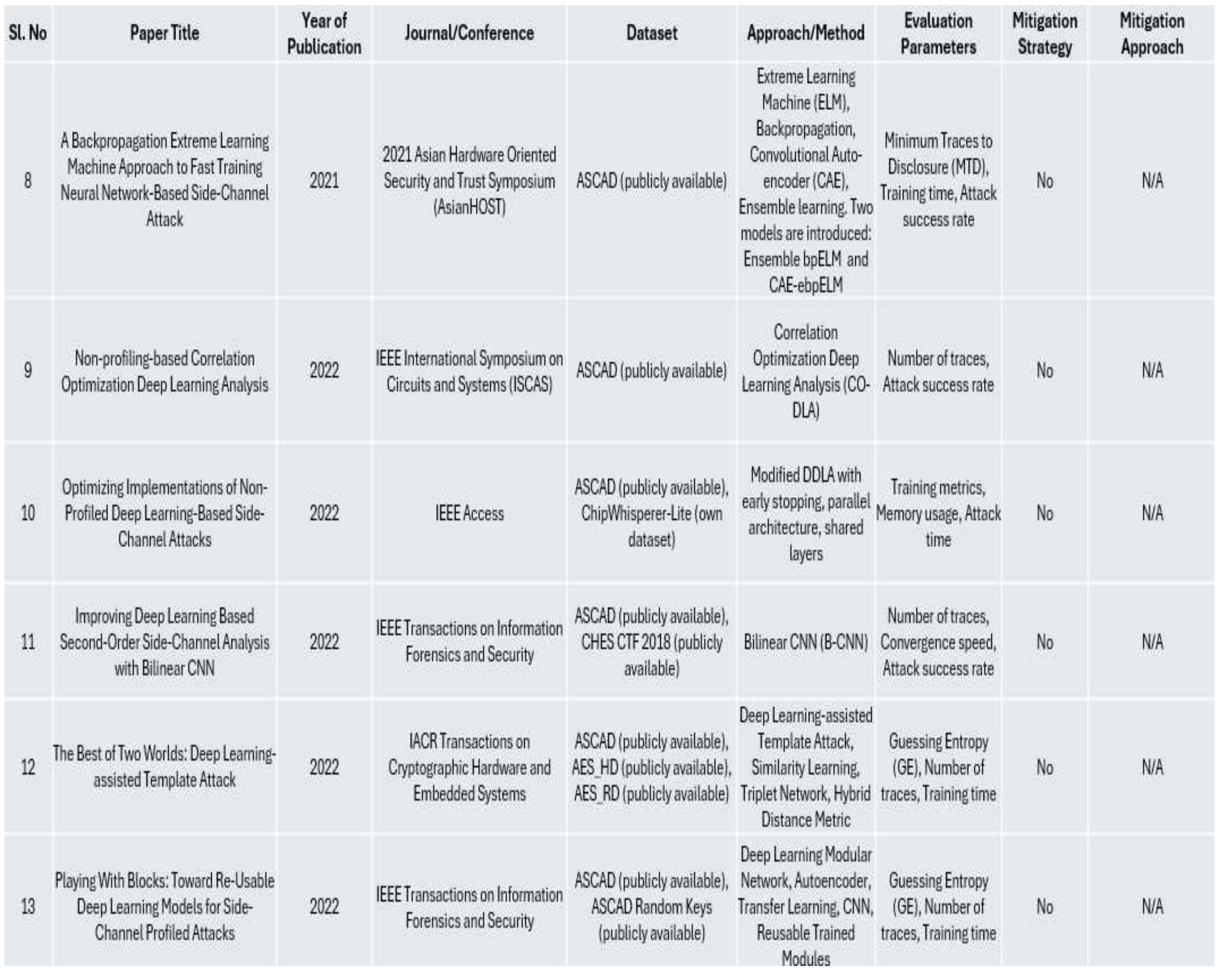
Sl. No	Paper Title	Year of Publication	Journal/Conference	Dataset	Approach/Method	Evaluation Parameters	Mitigation Strategy	Mitigation Approach
1	Make Some Noise: Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis	2019	IACR Transactions on Cryptographic Hardware and Embedded Systems	DPAcontest v4 (publicly available), AES_HD (publicly available), AES_RD (publicly available), ASCAD (publicly available)	Convolution Neural Network(CNN), Noise Addition, Regularization, Batch Normalization, Dropout, 1-D VGG	Guessing Entropy (GE), Number of traces, Noise levels, Training time	No	N/A
2	Deep learning for side-channel analysis and introduction to ASCAD database	2020	Journal of Cryptographic Engineering	ASCAD (publicly available)	Deep Learning, CNN, MultiLayer Perceptron(MLP)	Number of epochs, Batch size, Rank function, Accuracy, Training time	No	N/A
3	A Novel Evaluation Metric for Deep Learning-Based Side Channel Analysis and Its Extended Application to Imbalanced Data	2020	IACR Transactions on Cryptographic Hardware and Embedded Systems	ASCAD (publicly available), DPA Contest v4 (publicly available), AES_RD (publicly available), AES_HD (publicly available)	Cross Entropy Ratio (CER) metric, CER loss function	Guessing Entropy (GE), Success Rate (SR), Cross Entropy Ratio (CER)	Yes	Data balancing, use of CER loss function
4	Effective Deep Learning-based Side-Channel Analyses Against ASCAD	2021	IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)	ASCAD (publicly available)	CNN with heatmap and SNR reduction, Multi-label classification, Transfer learning	Number of traces, Training time	No	N/A
5	Towards Strengthening Deep Learning-based Side Channel Attacks with Mixup	2021	IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)	ASCAD (publicly available), ASCAD desync50 (publicly available), ASCAD desync100 (publicly available), AES_RD (publicly available)	Data Augmentation with Mixup, Correlation Power Analysis (CPA)	Number of traces, Attack success rate, Mixup parameter (α)	No	N/A
6	Leveraging Deep CNN and Transfer Learning for Side-Channel Attack	2021	22nd International Symposium on Quality Electronic Design (ISQED)	ASCAD (publicly available)	Deep 2-D CNN, CWT, Scalograms. GoogLeNet, InceptionV3, VGG16, MobileNetV2	Key rank, Accuracy, Number of traces	No	N/A
7	Multilabel Deep Learning-Based Side-Channel Attack	2021	IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems	ASCAD (publicly available), AES_HD (publicly available), AES_RD (publicly available)	Multilabel Classification, CNN, MLP Ensemble Learning	Guessing Entropy (GE), Number of traces, Model Complexity	No	N/A

Figure 2.5: Literature review on deep learning (DL) based side-channel attacks

strategies such as data balancing and the use of the CER loss function were implemented, focusing on GE and success rate (SR) [73].

- **Effective Deep Learning-Based Side-Channel Analyses Against ASCAD:** This work used the ASCAD dataset and applied CNN with heatmap and SNR reduction, multi-label classification, and transfer learning. Evaluation parameters included the number of traces and training time. No measures to combat side-channel attacks were implemented [74].

- **Towards Strengthening Deep Learning-Based Side-Channel Attacks with Mixup:** This study leveraged the ASCAD desync50 and desync100 datasets with data augmentation via Mixup and Correlation Power Analysis



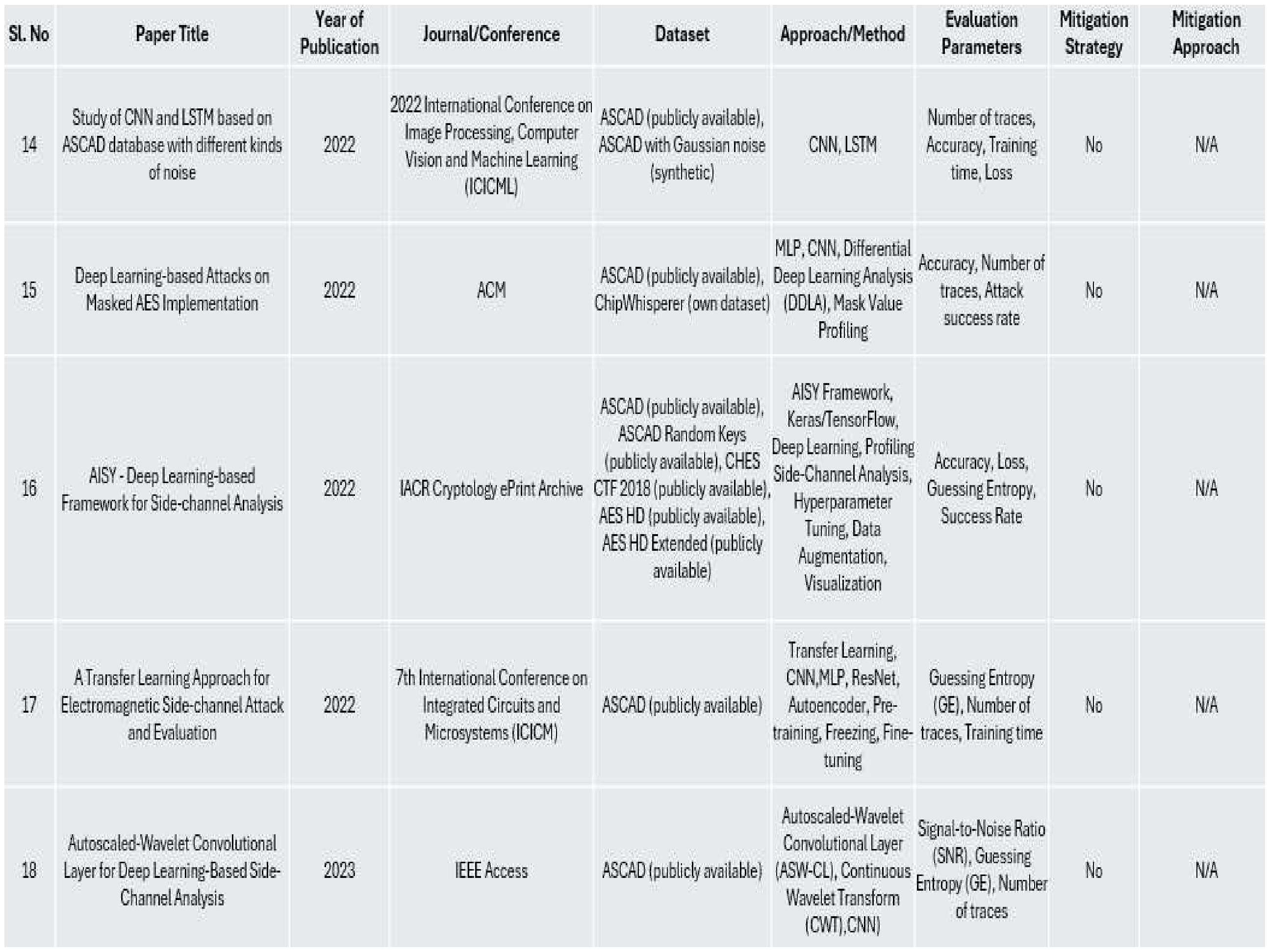
Sl. No	Paper Title	Year of Publication	Journal/Conference	Dataset	Approach/Method	Evaluation Parameters	Mitigation Strategy	Mitigation Approach
8	A Backpropagation Extreme Learning Machine Approach to Fast Training Neural Network-Based Side-Channel Attack	2021	2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)	ASCAD (publicly available)	Extreme Learning Machine (ELM), Backpropagation, Convolutional Auto-encoder (CAE), Ensemble learning. Two models are introduced: Ensemble bpELM and CAE-ebpELM	Minimum Traces to Disclosure (MTD), Training time, Attack success rate	No	N/A
9	Non-profiling-based Correlation Optimization Deep Learning Analysis	2022	IEEE International Symposium on Circuits and Systems (ISCAS)	ASCAD (publicly available)	Correlation Optimization Deep Learning Analysis (CO-DLA)	Number of traces, Attack success rate	No	N/A
10	Optimizing Implementations of Non-Profiled Deep Learning-Based Side-Channel Attacks	2022	IEEE Access	ASCAD (publicly available), ChipWhisperer-Lite (own dataset)	Modified DDLA with early stopping, parallel architecture, shared layers	Training metrics, Memory usage, Attack time	No	N/A
11	Improving Deep Learning Based Second-Order Side-Channel Analysis with Bilinear CNN	2022	IEEE Transactions on Information Forensics and Security	ASCAD (publicly available), CHES CTF 2018 (publicly available)	Bilinear CNN (B-CNN)	Number of traces, Convergence speed, Attack success rate	No	N/A
12	The Best of Two Worlds: Deep Learning-assisted Template Attack	2022	IACR Transactions on Cryptographic Hardware and Embedded Systems	ASCAD (publicly available), AES_HD (publicly available), AES_RD (publicly available)	Deep Learning-assisted Template Attack, Similarity Learning, Triplet Network, Hybrid Distance Metric	Guessing Entropy (GE), Number of traces, Training time	No	N/A
13	Playing With Blocks: Toward Re-Usable Deep Learning Models for Side-Channel Profiled Attacks	2022	IEEE Transactions on Information Forensics and Security	ASCAD (publicly available), ASCAD Random Keys (publicly available)	Deep Learning Modular Network, Autoencoder, Transfer Learning, CNN, Reusable Trained Modules	Guessing Entropy (GE), Number of traces, Training time	No	N/A

Figure 2.6: Literature review on deep learning (DL) based side-channel attacks

(CPA). Evaluation parameters included the number of traces, attack success rate, and Mixup parameter. No mitigation strategy was employed [75].

- **Leveraging Deep CNN and Transfer Learning for Side-Channel Attack:** This study utilized ASCAD, AES_RD, and AES_HD datasets, employing deep 2-D CNN, Continuous Wavelet Transform (CWT), scalograms, and models like GoogLeNet, InceptionV3, VGG16, and MobileNetV2. Evaluation parameters included key rank, accuracy, and the number of traces. No mitigation strategy was mentioned [76].

- **Multilabel Deep Learning-Based Side-Channel Attack:** This research utilized the ASCAD dataset and leveraged multi-label deep learning models.



Sl. No	Paper Title	Year of Publication	Journal/Conference	Dataset	Approach/Method	Evaluation Parameters	Mitigation Strategy	Mitigation Approach
14	Study of CNN and LSTM based on ASCAD database with different kinds of noise	2022	2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)	ASCAD (publicly available), ASCAD with Gaussian noise (synthetic)	CNN, LSTM	Number of traces, Accuracy, Training time, Loss	No	N/A
15	Deep Learning-based Attacks on Masked AES Implementation	2022	ACM	ASCAD (publicly available), ChipWhisperer (own dataset)	MLP, CNN, Differential Deep Learning Analysis (DDLA), Mask Value Profiling	Accuracy, Number of traces, Attack success rate	No	N/A
16	AISY - Deep Learning-based Framework for Side-channel Analysis	2022	IACR Cryptology ePrint Archive	ASCAD (publicly available), ASCAD Random Keys (publicly available), CHES CTF 2018 (publicly available), AES HD (publicly available), AES HD Extended (publicly available)	AISY Framework, Keras/TensorFlow, Deep Learning, Profiling Side-Channel Analysis, Hyperparameter Tuning, Data Augmentation, Visualization	Accuracy, Loss, Guessing Entropy, Success Rate	No	N/A
17	A Transfer Learning Approach for Electromagnetic Side-channel Attack and Evaluation	2022	7th International Conference on Integrated Circuits and Microsystems (ICICM)	ASCAD (publicly available)	Transfer Learning, CNN,MLP, ResNet, Autoencoder, Pre-training, Freezing, Fine-tuning	Guessing Entropy (GE), Number of traces, Training time	No	N/A
18	Autoscaled-Wavelet Convolutional Layer for Deep Learning-Based Side-Channel Analysis	2023	IEEE Access	ASCAD (publicly available)	Autoscaled-Wavelet Convolutional Layer (ASW-CL), Continuous Wavelet Transform (CWT),CNN)	Signal-to-Noise Ratio (SNR), Guessing Entropy (GE), Number of traces	No	N/A

Figure 2.7: Literature review on deep learning (DL) based side-channel attacks

The evaluation focused on GE, the number of traces, model hyperparameters, and training time. No mitigation strategies were implemented [77].

- **A Backpropagation Extreme Learning Machine Approach to Fast Training Neural Network-Based Side-Channel Attack:** This study made use of the ASCAD dataset and examined Extreme Learning Machine (ELM), backpropagation, and Convolutional Autoencoder (CAE) with ensemble learning. It introduced two models: Ensemble bpELM and CAE-bpELM. Evaluation parameters used comprised Minimum Traces to Disclosure (MTD), training time, and attack success rate. There was no mitigation strategy applied [78].

- **Non-Profiling-Based Correlation Optimization Deep Learning Analysis:** This research utilized the ASCAD dataset and used techniques such as Correlation Optimization Deep Learning Analysis (CO-DLA). The evaluation parameters dealt with the number of traces and attack success rate. No mitigation strategy was mentioned [79].

- **Optimizing Implementations of Non-Profiled Deep Learning-Based Side-Channel Attacks:** This research leveraged a combination of three datasets namely publicly available ASCAD, ChipWhisperer-Lite, and their own dataset. They employed Modified Deep Learning-Based Differential Learning Analysis (DDLA) with early stopping, parallel architecture, and shared layers. Evaluation metrics consisted of training metrics, memory usage, and attack time. There is no countermeasure [80].

- **Improving Deep Learning-Based Second-Order Side-Channel Analysis with Bilinear CNN:** This work used the dataset ASCAD and CHES CTF 2018 datasets. It demonstrated the power of Bilinear CNN (B-CNN) and explored various evaluation parameters such as the number of traces, convergence speed, and attack success rate. No mitigation strategy was provided [81].

- **The Best of Two Worlds: Deep Learning-Assisted Template Attack:** This analysis made use of ASCAD, AES_HD, and AES_RD datasets. It employed deep learning-assisted template attacks with similarity learning, triplet networks, and hybrid distance metrics. The evaluation parameters were guessing entropy (GE), the number of traces, and training time. Mitigation strategies were not applied [82].

- **Playing With Blocks: Toward Re-Usable Deep Learning Models for Side-Channel Profiled Attacks:** This research used ASCAD, AES_HD, and a dataset with random keys. It explored deep learning modular networks, autoencoders, transfer learning, and reusable training modules. Evaluation parameters included GE, the number of traces, and training time. No mitigation strategy was mentioned [83].

- **Study of CNN and LSTM based on ASCAD database with different kinds of noise:** This study utilized the ASCAD dataset with synthetic Gaussian noise, employing CNN and LSTM models. The evaluation parameters were the number of traces, accuracy, training time, and loss. No mitigation strategies were implemented in this study [84].

- **Deep Learning-based Attacks on Masked AES Implementation:** This paper used the publicly available ASCAD and ChipWhisperer datasets. The study leveraged Multi-Layer Perceptron (MLP), CNN, Differential Deep Learning Analysis (DDLA), and Mask Value Profiling. The evaluation parameters were accuracy, number of traces, and attack success rate. No mitigation techniques were employed [85].

- **AISY - Deep Learning-based Framework for Side-channel Analysis" (2022):** This study utilized multiple datasets, namely ASCAD, ASCAD Random Keys, CHES CTF 2018, AES HD, and AES HD Extended. They extended the AISY Framework with Keras/TensorFlow, focusing on deep learning, profiling side-channel analysis, hyperparameter tuning, data augmentation, and visualization. Evaluation parameters included accuracy, loss, guessing entropy, and success rate, without any mitigation strategies [86].

- **A Transfer Learning Approach for Electromagnetic Side-channel Attack and Evaluation:** This study used the ASCAD dataset and applied techniques such as transfer learning, CNN, MLP, ResNet, Autoencoder, pre-training, freezing, and fine-tuning. Evaluation parameters included guessing entropy (GE), number of traces, and training time, with no mitigation strategies mentioned [87].

- **Autoscaled-Wavelet Convolutional Layer for Deep Learning-Based Side-Channel Analysis:** This study utilized the ASCAD dataset and applied Autoscaled-Wavelet Convolutional Layer (ASW-CL), Continuous Wavelet Transform (CWT), and CNN. The evaluation parameters were signal-to-noise ratio (SNR), guessing entropy (GE), and the number of traces. No countermeasures were employed in this paper [88].

2.2.3 Summary of Literature Survey

In the literature survey on Differential Power Analysis (DPA) and Correlation Power Analysis (CPA), the focus is primarily on improving statistical methods to effectively analyze vulnerabilities in cryptographic systems through power trace analysis. DPA involves collecting many power traces and statistically analyzing them to detect patterns that reveal cryptographic keys. CPA builds on this by using correlation coefficients to better match predicted power consumption with actual measurements, thus refining the accuracy in deducing keys.

The evaluation metrics for DPA and CPA focus on the success rate, the number of traces required to successfully deduce information, and how well these methods perform in the presence of noise and countermeasures. These metrics are crucial in developing statistical models that can perform well even with fewer data points or under interference, which is essential for their practical application.

On the other hand, research involving Deep Learning (DL) methodologies for side-channel attacks is centered around using neural networks like CNNs and RNNs. These models automate the process of extracting useful features from power traces, which minimizes the need for extensive domain knowledge and allows for direct processing of raw data, greatly improving the efficiency of the analysis.

For DL approaches, the evaluation metrics include model accuracy, robustness against noisy data, and the ability to adapt across different cryptographic algorithms and datasets. Additionally, the computational efficiency of these models is also considered due to the significant resources required to train complex neural architectures, affecting their practical deployment.

In summary, both traditional and DL-based methods are evolving to incorporate more advanced statistical and machine learning tools to better address challenges in cryptographic security. This progress not only enhances the effectiveness of side-channel attacks but also aids in the development of more effective defenses against such threats.

2.2.4 Research gap

The identified research gap in the existing literature primarily revolves around the absence of robust mitigation strategies against side-channel attacks (SCAs) and limited exploration of advanced deep-learning techniques to enhance security. Numerous studies rely heavily on conventional deep learning models like CNNs and MLPs without incorporating effective countermeasures against SCAs.

While some papers introduce novel evaluation metrics or hybrid techniques, there remains a substantial gap in applying these sophisticated methods extensively across various datasets or in tandem with robust mitigation measures. Furthermore, these studies typically restrict their evaluations to standard datasets like ASCAD, overlooking the potential for these methods to provide broader defense mechanisms across different cryptographic scenarios.

Additionally, there is a notable absence in varying the number of power traces used in training and evaluation, which is crucial for understanding how these models perform under data-constrained conditions. This oversight indicates an opportunity to expand research to include a wider array of conditions and attack types, ensuring that findings are robust and applicable in real-world scenarios where data availability may vary.

Finally, a notable research deficiency exists in the classification of power traces and the generation of simulated or synthetic power traces, which are essential for developing adaptable and resilient cryptographic defenses. The absence of comprehensive studies that classify power traces or create their synthetic equivalents suggests an area for further exploration and development. These studies highlight the need for advanced techniques and strategies to improve the resilience of AES implementations against side-channel attacks.

2.2.5 Novelty

Our research introduces several innovative approaches to enhance security analysis and key deduction capabilities.

- **Generation and analysis of synthetic power traces:** We develop a method to create synthetic power traces, which allows for a controlled study of side-channel attack vectors without the need for actual hardware.

- **Classification of AES implementations:** By analyzing power traces, we categorize different AES implementations based on distinctive features extracted during cryptographic operations. This classification helps in identifying vulnerable AES configurations.

- **Traces retrieval through Hashing and Metadata:** Utilize advanced hashing and metadata techniques to uniquely retract the original power traces from processed datasets. This capability is crucial for verifying the integrity of data and ensuring the reproducibility of our analysis.

- **Key deduction using GRUTrace and ResTraceNet Models with 100 power traces:** We employ two novel deep learning architectures, GRUTrace and ResTraceNet that are designed to capture and exploit temporal and residual patterns in power trace data, significantly improving the precision of key deduction. Our models aim to demonstrate their efficiency by deducing keys from as few as 100 power traces, highlighting their potential for high-accuracy performance even under data-limited conditions.

- **Mitigation strategies:** Our research focused on employing a mitigation strategy involving structured masking and adding Gaussian noise to protect AES algorithm against SCAs.

Chapter 3

Module 1: Generation And Analysis of Power Traces

The focus of Module 1 was on collecting and analyzing power traces from a masked AES implementation. The primary objective was to gather real power traces using an emulator setup and to generate synthetic power traces with a Python script based on the Hamming Weight (HW) power model. Various techniques were then employed to align and analyze both the real and synthetic power traces to validate the accuracy and reliability of the generated synthetic traces. This module is crucial in determining the feasibility of using synthetic data to augment real training data (raw power traces) in side-channel attacks analysis, which is essential in scenarios where obtaining real power traces is challenging or impossible. Figure 3.1 provides an overview highlighting Module 1 of the research contributions.

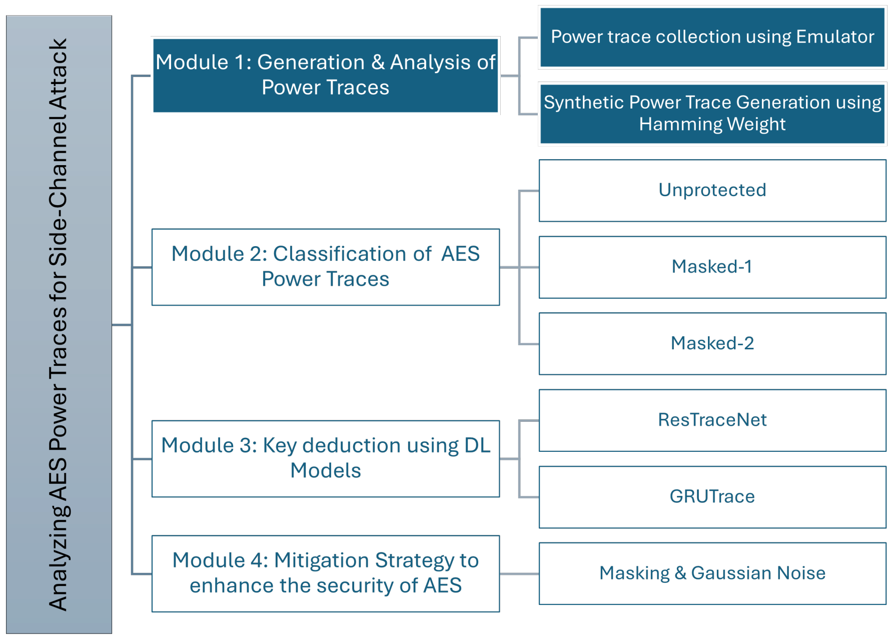

Analyzing AES Power Traces for Side-Channel Attack
Module 1: Generation & Analysis of Power Traces
Power trace collection using Emulator
Synthetic Power Trace Generation using Hamming Weight
Module 2: Classification of AES Power Traces
Unprotected
Masked-1
Masked-2
Module 3: Key deduction using DL Models
ResTraceNet
GRUTrace
Module 4: Mitigation Strategy to enhance the security of AES
Masking & Gaussian Noise

Figure 3.1: Research Contributions: Highlighted Module 1

3.1 Methodology

This section provides a comprehensive account of the methodologies employed to collect and analyze the power traces from a masked AES implementation. The methodologies are classified into three main sections:

- Collection of power traces using an emulator setup
- Generation of power traces using a Hamming weight (HW) based Python script
- Analyzing, aligning, and correlating the power traces using various techniques

3.1.1 Power trace collection using an Emulator

The masked AES algorithm was implemented on an Arduino UNO microcontroller. Masked AES implementation enhances resistance against SCAs by introducing randomness into the encryption process [89]. The AES-masked code was developed and executed with small modifications to incorporate masking, using the Arduino Integrated Development Environment (IDE) on Serial Port 4 to ensure efficient implementation and functionality. This modification ensures that each execution of the algorithm yields a varied power consumption profile, making it more difficult for attackers to infer information from these traces. This setup is critical for providing an environment where all the operations in AES can be monitored to generate accurate power traces at a frequency of 1 MHz, which is suitable for capturing variations in power consumption patterns during encryption. Fig. 3.2 shows the data collection setup used for collecting power traces.

Setup:

The power trace collection setup for analyzing the masked AES algorithm requires an extensive array of equipment to ensure precise implementation and collection of the generated power traces from the Arduino UNO. The core of this setup is the Syscomp CGR-101, a compact digital oscilloscope and waveform generator. This powerful, general-purpose signal generator is portable and user-friendly, with a maximum bandwidth of 2 MHz and a real-time sampling rate of 20 Mega Samples/second, featuring a 3.2-inch LED display. It offers 20 MS/s dual-channel, 10-bit oscilloscope capabilities, a 2 MHz DDS arbitrary waveform generator, and eight-channel digital I/O. It is USB powered, requiring no AC adaptor, and comes with open-source software compatible with Windows, Linux, and Mac. Power is supplied to the Arduino, which performs the encryption operation, using a 9V battery [90]. A 220-ohm resistor is used in the circuit setup to regulate the current flow and control the power input to the Arduino. This setup is crucial for providing an environment where all AES operations can be monitored to generate accurate power traces at a frequency of 1 MHz, appropriate for capturing variations in power consumption patterns during encryption. Power traces were collected exclusively from Channel A, while Channel B was disabled. The amplitude, frequency, and time base were adjusted to obtain accurate and reliable readings.

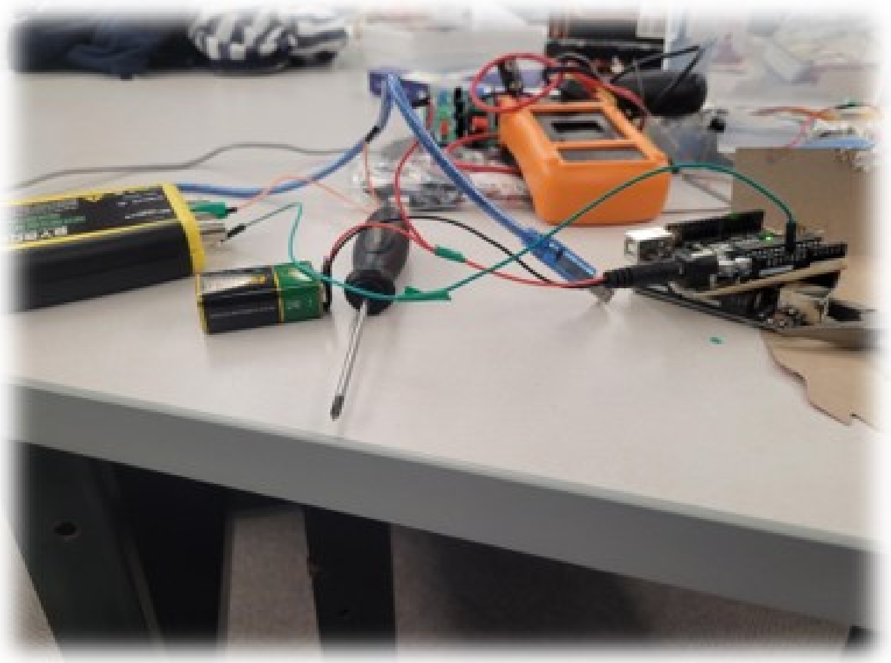


Figure 3.2: Data collection setup in a masked AES implementation

Data collection:

Power traces were collected at specific points during the execution of the cryptographic operations to provide maximum coverage of all stages of the AES algorithm. These points of interest were strategically chosen to capture the different phases, offering a detailed view of the power traces. They were collected during the following

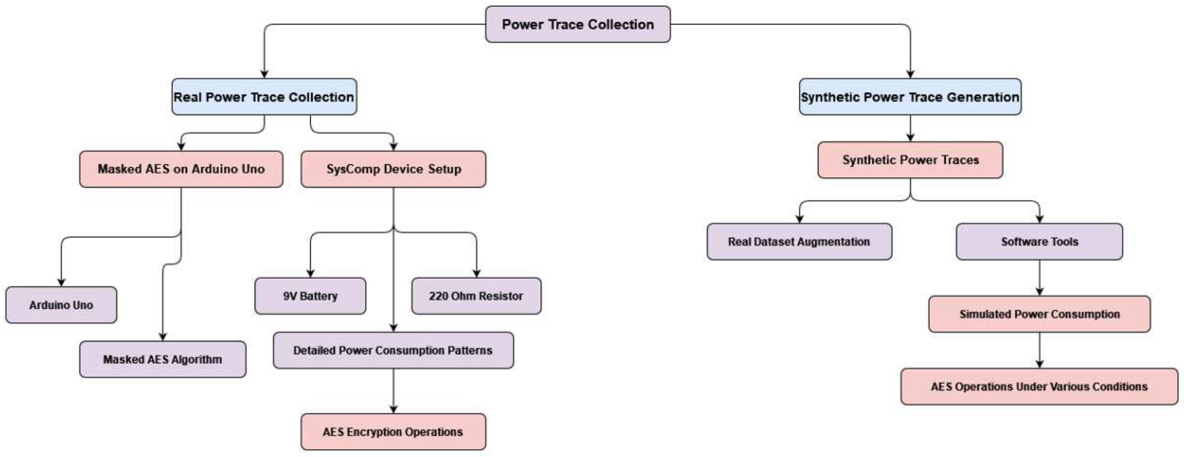

Power Trace Collection
Real Power Trace Collection
Synthetic Power Trace Generation
Masked AES on Arduino Uno
SysComp Device Setup
Synthetic Power Traces
Real Dataset Augmentation
Software Tools
Arduino Uno
Masked AES Algorithm
9V Battery
220 Ohm Resistor
Detailed Power Consumption Patterns
AES Encryption Operations
Simulated Power Consumption
AES Operations Under Various Conditions

Figure 3.3: Power traces collection method

phases: Fig 3.3 shows both the power collection methods.

- **Initialization Phase:** Power traces were collected as the initial key and state matrix were set up, serving as a baseline for power trace collection.

- **SubBytes Phase:** Power consumption was monitored as each byte in the state matrix was substituted using the S-box, introducing nonlinearity into the process.

- **ShiftRows Phase:** Power traces were collected during the cyclic left shift of each row in the state matrix, analyzing the effect of transposition on power consumption. This step is crucial for providing diffusion, an integral security feature of the algorithm.

- **MixColumns Phase:** Power traces were gathered when the columns of the state matrix were mixed and multiplied by a fixed polynomial, further enhancing diffusion within the algorithm.

- **AddRoundKey Phase:** Power consumption was recorded as the state matrix was combined with the round key using a bitwise XOR operation, introducing confusion. This phase is particularly crucial as it directly involves the cryptographic key.

- **Final Round:** In the final round, traces were collected during all operations except MixColumns, ensuring that the final transformation of the state matrix was captured comprehensively.

Data Analysis:

The data analysis process leverages the Syscomp CircuitGear 1.24 software, which collects and computes the voltage drop across the resistor to calculate the power traces during the execution of certain cryptographic operations in a masked AES algorithm implementation. From the collected measurements, a detailed dataset was created using the settings provided in the software that converted the power traces into a CSV file. This dataset encapsulates all the variations in power consumption, enabling the evaluation of masking's effectiveness in concealing potential data leakage. The dataset is crucial for determining whether sensitive information can be deduced from these raw power traces and, if so, for developing stronger countermeasures to enhance cryptographic security. Fig. 3.4 shows the Syscomp CircuitGear 1.24 software.

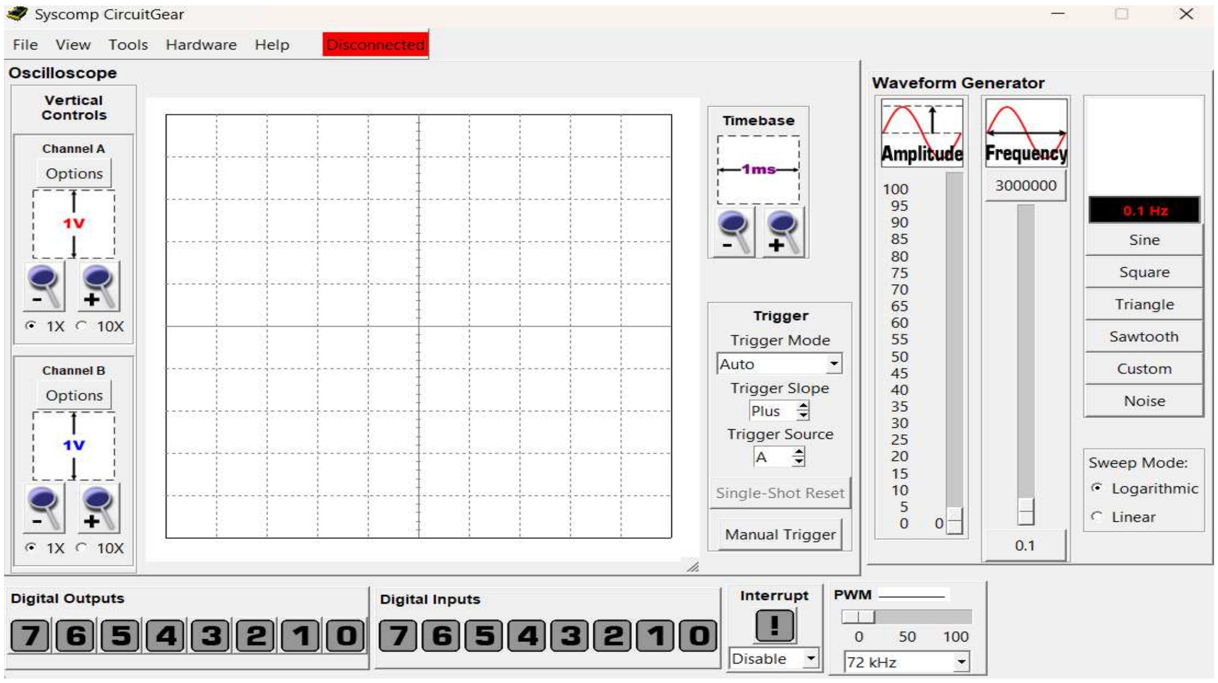


Figure 3.4: Syscomp CircuitGear 1.24 software

Mathematical Derivation of Power Consumption for Masked AES Algorithm Using Oscilloscope Measurements

We derive a mathematical equation for the power analysis collection setup, for which we take into account the collected power traces from the Arduino UNO running a masked AES algorithm. The crux of the implementation consists of a Syscomp CGR-101 oscilloscope and waveform generator, which has specific capabilities and configurations. Fig. 3.5 shows the visual representation of the power traces collection along with the mathematical components needed to derive the power consumption.

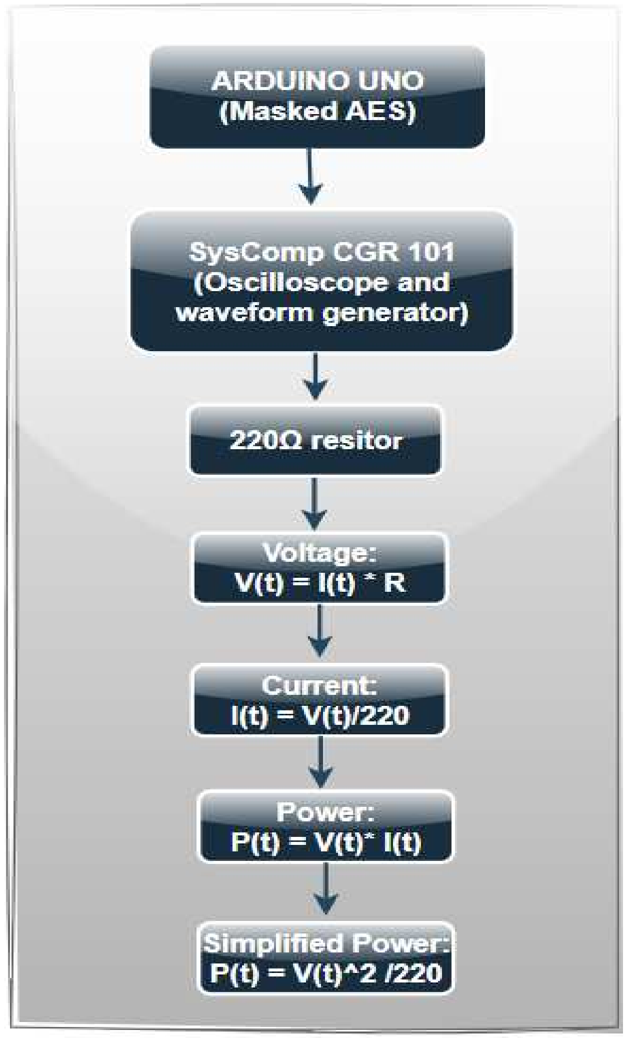


Figure 3.5: Power Analysis Collection Setup for Arduino UNO Running Masked AES Algorithm

Step 1: Define the Setup

- **Oscilloscope Specifications:**

  - Bandwidth: 2 MHz
  - Sampling Rate: 20 MSa/s
  - Dual-channel, 10-bit resolution (only Channel A is used)
  - Power Supply: USB, powered by a 9V battery

- **Resistor in Circuit:** $R = 220\,\Omega$
- **Frequency for Power Traces:** 1 MHz

Step 2: Voltage and Current Relationship

The voltage $V(t)$ across the 220-ohm resistor is directly related to the current $I(t)$ flowing through the Arduino circuit as per ohms law.

$$V(t) = I(t) \cdot R \tag{3.1}$$

Given $R = 220\,\Omega$, we have:

$$V(t) = 220 \cdot I(t) \tag{3.2}$$

Step 3: Power Consumption Calculation

The instantaneous power $P(t)$ consumed by the Arduino can be calculated using:

$$P(t) = V(t) \cdot I(t) \tag{3.3}$$

Substitute $V(t) = 220 \cdot I(t)$:

$$P(t) = (220 \cdot I(t)) \cdot I(t) \tag{3.4}$$

$$P(t) = 220 \cdot I(t)^2 \tag{3.5}$$

Step 4: Sampling and Data Collection

The oscilloscope samples the voltage at a rate of 20 MSa/s. Let V_i be the sampled voltage at the i-th sample. The corresponding instantaneous current I_i is:

$$I_i = \frac{V_i}{220} \tag{3.6}$$

The instantaneous power at the i-th sample is then:

$$P_i = 220 \cdot \left(\frac{V_i}{220}\right)^2 \tag{3.7}$$

$$P_i = \frac{V_i^2}{220} \tag{3.8}$$

Step 5: Frequency Adjustment

The power traces have been collected at a frequency of 1 MHz, which allows us to examine the power consumption for each cycle of the AES algorithm. It's important to analyze the oscilloscope data to ensure it matches the 1 MHz frequency.

Step 6: Mathematical Representation

We derive the mathematical equation for the power traces collected by combining the steps above,

$$P_i = \frac{V_i^2}{220} \tag{3.9}$$

where V_i is the voltage measured at the i-th sample by the oscilloscope.

The power consumption P_i at each sample point i can be calculated using the equation:

$$P_i = \frac{V_i^2}{220} \tag{3.10}$$

The equation here represents the connection between the voltage recorded by the oscilloscope and the power usage of the Arduino executing the AES algorithm. Accurate analysis of power traces requires careful consideration of the oscilloscope's specifications and the circuit configuration.

3.1.2 Generation of synthetic power traces

The primary objective is to programmatically generate power traces that can accurately mimic the raw traces captured during the execution of a masked AES algorithm in a controlled environment. As described in the previous section, AES is implemented with an additional masking mechanism to obscure the relationship between cryptographic operations and power consumption data. The implemented Python script performs AES encryption operations by applying masking at various phases, ensuring that the power consumption profile obtained is distinct for each phase. Emulating a real-world scenario where an attacker attempts to deduce key information, this step is crucial for understanding and mitigating potential vulnerabilities.

Data collection:

During each encryption cycle iteration, the code logs various key pieces of information:

- **Power Traces:** Generated based on the Hamming Weight model.
- **Plaintext:** The original data before encryption.
- **S-box Output:** Results from the substitution box operation that provides non-linearity.
- **Ciphertext:** The encrypted output.
- **Key:** The encryption key used in that specific cycle.

The primary purpose of collecting this information is to generate a detailed dataset that illustrates how the power trace profiles vary based on the cryptographic operations being performed. This comprehensive dataset is crucial for aligning and comparing the synthetic traces with the real traces during the analysis phase. By doing so, it helps validate the accuracy and reliability of the synthetic traces in mimicking real-world power consumption patterns.

Hamming Weight(HW) model:

The power consumption of the masked AES implementation during the encryption process is correlated to the Hamming Weight (HW) of the data's state. The HW measures the number of ones in the input data, which correlates with the power consumed by a digital circuit when changing state from 0 to 1 [91]. For every byte of input processed, the Python script computes an HW value, which is then used to generate the power trace measurement for that operation. This process simulates the actual measurements observed during real power trace collection. The Hamming weight HW[x] is the number of bits in x that are 1.

$$HW[x] = \sum_{i=1}^{n} x_i \tag{3.11}$$

The primary objective of this research is to generate a synthetic dataset that can simulate real power traces in a controlled and repeatable environment. The implications of this synthetic dataset are multifold. It can be augmented with training data for machine learning or deep learning models to recognize vulnerabilities or evaluate the efficacy of the implemented masking techniques. Additionally, it can validate side-channel analysis in scenarios where collecting real power traces and setting up the real environment is expensive and cumbersome. This approach also allows for the identification of potential weaknesses in the cryptographic implementation under various attack scenarios.

3.1.3 Analysis of the power traces

The collected raw power traces and synthetic traces are subjected to various techniques to analyze and find the correlation between the two types of traces. These techniques help us comprehend the behavior of the collected power traces, their relationships with cryptographic processes, and validate the reliability of the synthetically generated power traces. We apply eight techniques:

Fast-Fourier Transform (FFT) converts time-domain signals into frequency-domain signals, aiding in understanding periodic noise and components in the power traces[92]. Principal Component Analysis (PCA), a dimensionality reduction technique, transforms the original variables into principal components that retain integral features while reducing data complexity. Independent Component Analysis (ICA) explores the maximum statistical independence of the principal components, helping to segregate overlapping signal sources and isolate specific parts of the encryption process [93].

Cross-correlation analysis computes the similarity between the real and synthetic power traces based on their relative displacement [94]. Dynamic Time Warping (DTW) analyzes the similarity between two temporal sequences that differ in speed, aligning the real and synthetic power traces for accurate comparison. The wavelet transform performs a time-frequency analysis of the power traces for non-stationary signals whose frequency varies over time, identifying transient features [95].

Linear regression establishes a linear relationship between the dependent and independent variables, predicting real power traces based on synthetic traces to validate efficiency. Lastly, Random Forest, an ensemble method, is used for classification and regression, predicting or classifying behavior based on features derived from the power traces, thus offering insights into how these features correlate with different cryptographic states or operations [96].

3.2 Results and Discussion

The primary objective of this section is to discuss the results of the power trace collection and analysis, focusing on both real and synthetically generated power traces.

Real Power Trace Collection

In the first section, we successfully implemented the masked AES algorithm using the Arduino IDE on an Arduino UNO microcontroller. We collected the power traces using the Syscomp CGR-101 oscilloscope. These power traces provided a baseline for understanding how power consumption patterns correlate with each cryptographic operation. This baseline is essential for comparing and validating the synthetic power traces.

Synthetic Power Trace Generation

In the second section, we generated synthetic power traces programmatically based on the Hamming Weight (HW) model using a Python script. These synthetic power traces, as shown in Fig. 3.6 simulate the actual power traces collected in the first step. The comparison of the synthetic and real power traces is illustrated in the Encryption Step vs. Power Consumption and Frequency vs. Amplitude graphs as in Fig. 3.7

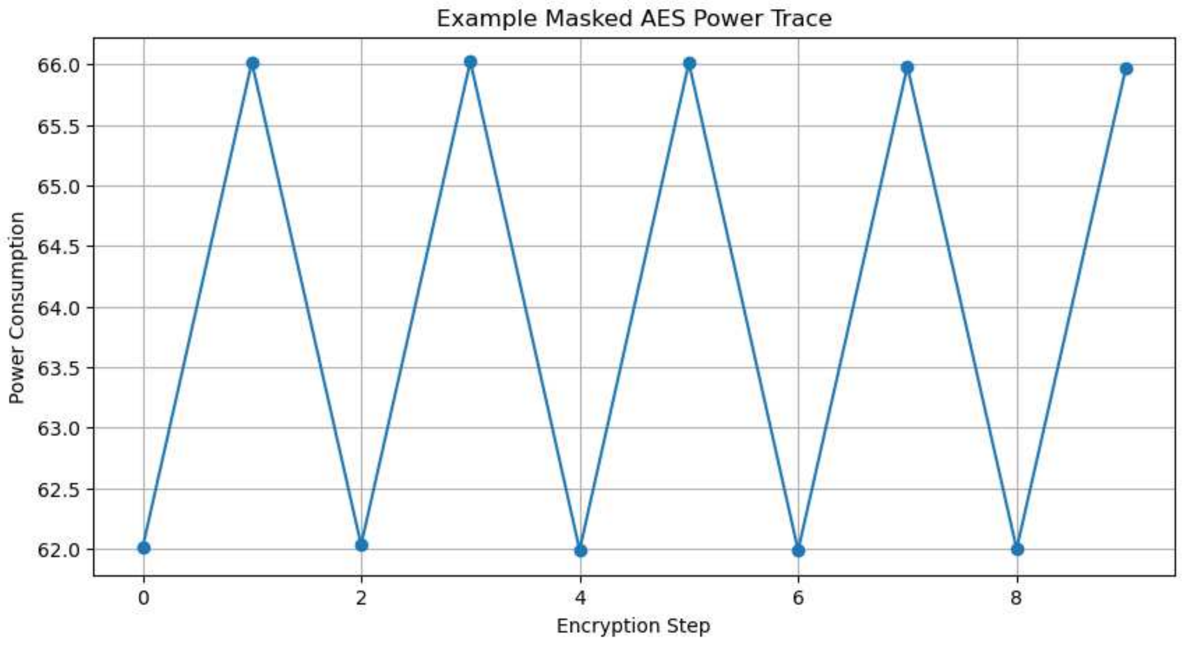


Figure 3.6: The synthetically generated power traces

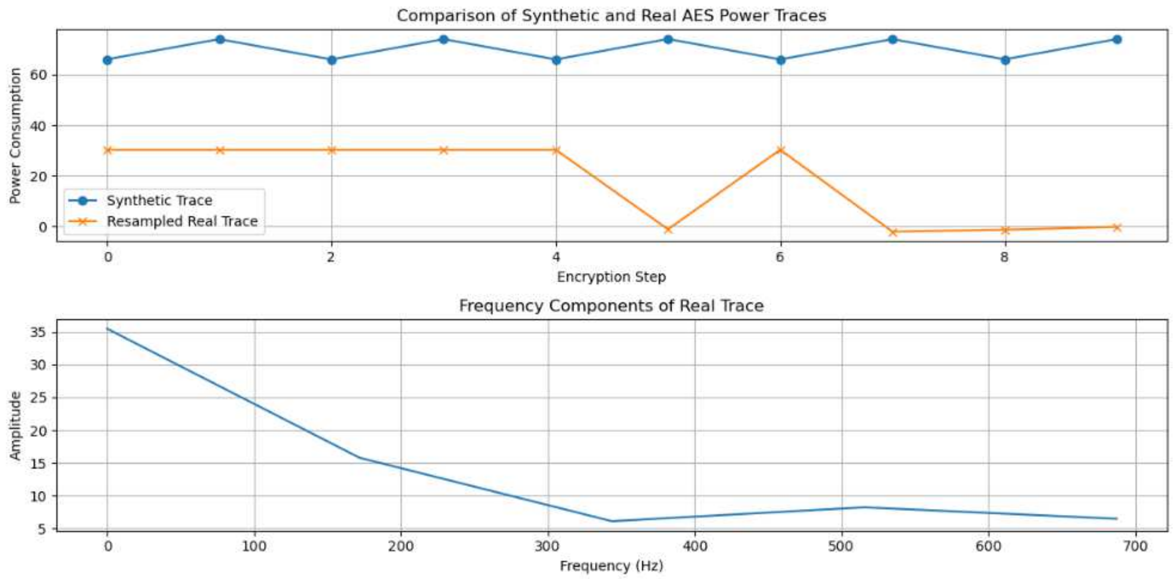


Figure 3.7: Comparison of synthetic and real power traces

Filtering and Smoothing

To improve the quality of the synthetic power traces, we applied filtering and smoothing techniques. The resulting traces, shown in Fig. 3.8 highlight the synthetic

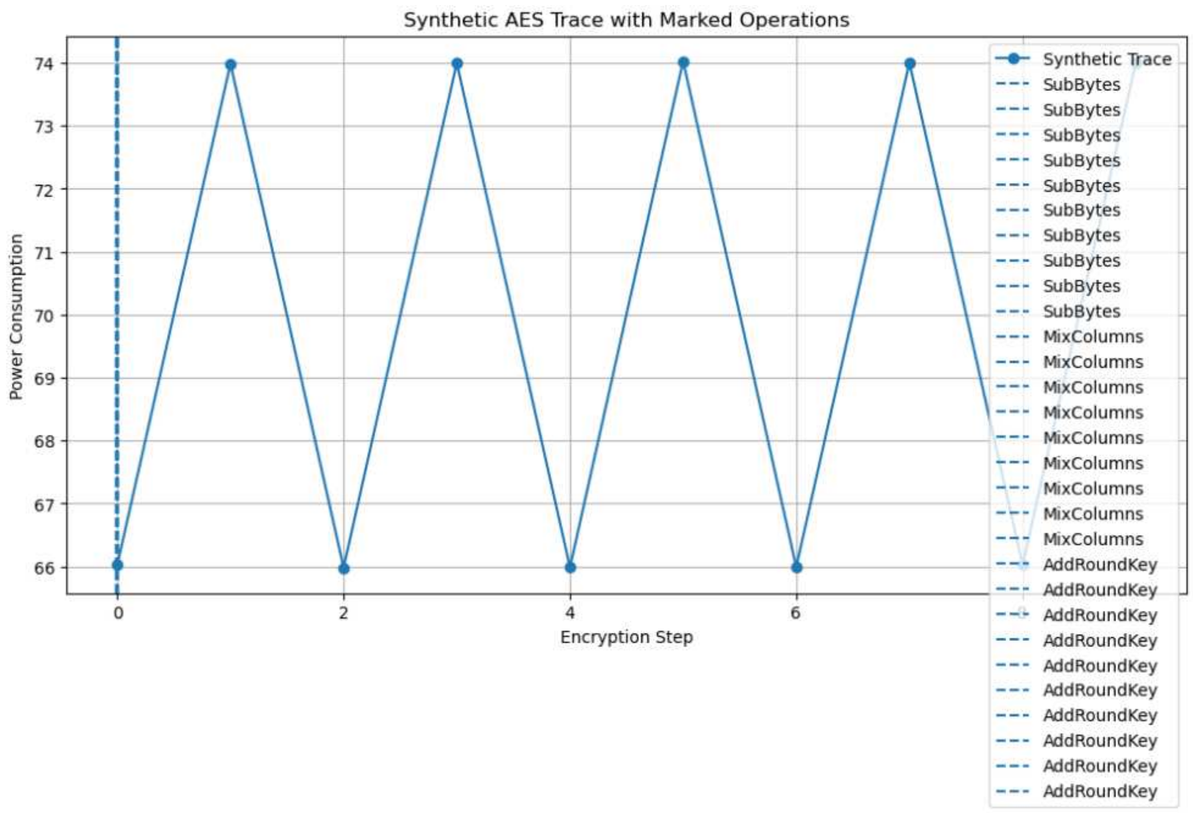


Figure 3.8: Comparison of real and synthetic power trace after smoothing and filtering

and real power traces after applying these techniques. Filtering and smoothing are essential for reducing noise and enhancing the quality of the data for further analysis.

Timing Analysis

A detailed timing analysis was performed on the real and synthetic power traces, as indicated in Fig. 3.9. This analysis explored the temporal aspects of the power consumption patterns to align the real and synthetic traces, achieving a high level of correlation between the two. The figures also depict the marked operations such as SubBytes, MixColumns, and AddRoundKey.

Techniques Applied for Analysis and Alignment of Real and Synthetic Power Traces

Using Fast-fourier transform (FFT), as shown in Fig. 3.10 we compared the synthetic power traces with the smoothed real power traces. The correlation coefficient

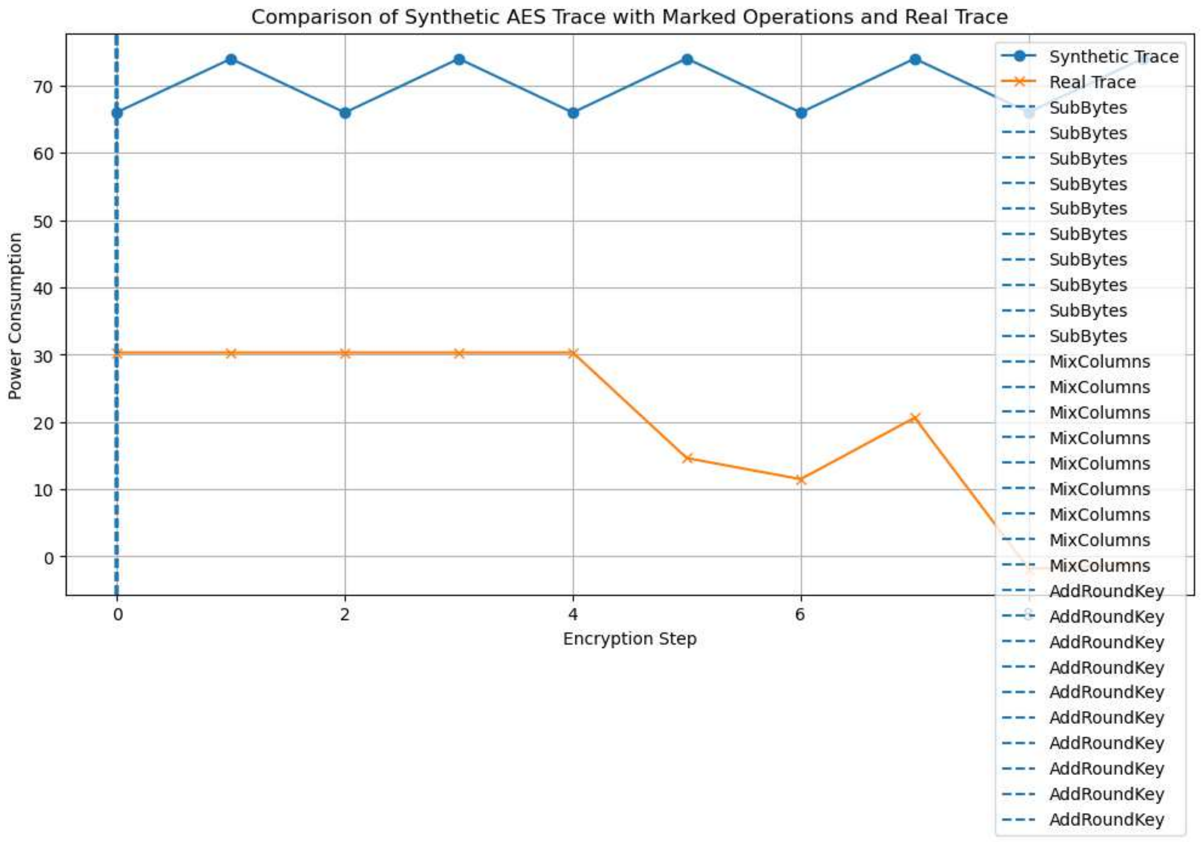


Figure 3.9: Timing analysis of the collected power traces

for this comparison is 0.1967, indicating a lower degree of similarity compared to the raw traces.

The correlation between the synthetic traces and Principal component analysis(PCA)-transformed real traces is shown in Fig. 3.11. The PCA technique reduced the dimensionality of the data while retaining the most significant features, resulting in a correlation coefficient of 0.3087. This moderate correlation indicates that PCA is useful in capturing the primary variance in the power consumption data, albeit with some loss of finer details.

The correlation between the synthetic traces and Independent component analysis(ICA) -transformed real traces is shown in Fig. 3.12. The ICA technique aimed to maximize the statistical independence of the components, resulting in a negative correlation coefficient of -0.3087. This indicates a divergence in the patterns captured

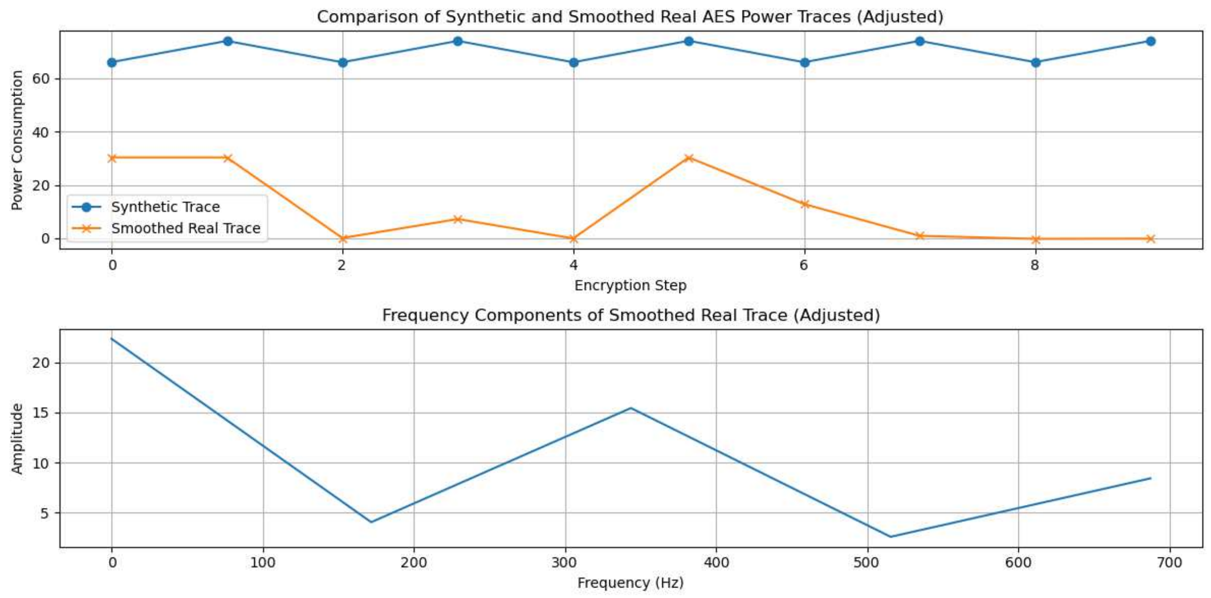


Figure 3.10: Comparison using Fast-fourier transform

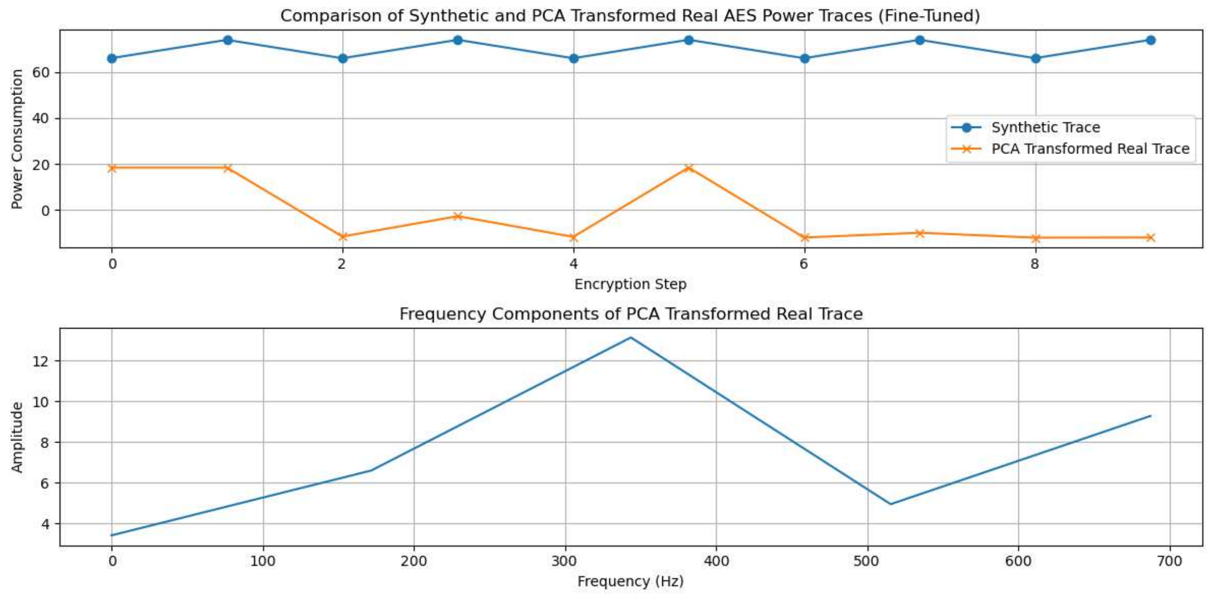


Figure 3.11: Comparison using Principal component analysis

by ICA, suggesting that while it can isolate components, the synthetic traces may not perfectly align with the independent components of the real traces.

The cross-correlation between synthetic and real traces, depicted in Fig. 3.13 revealed a maximum correlation value of 7404.856 at a lag of 2. This high value

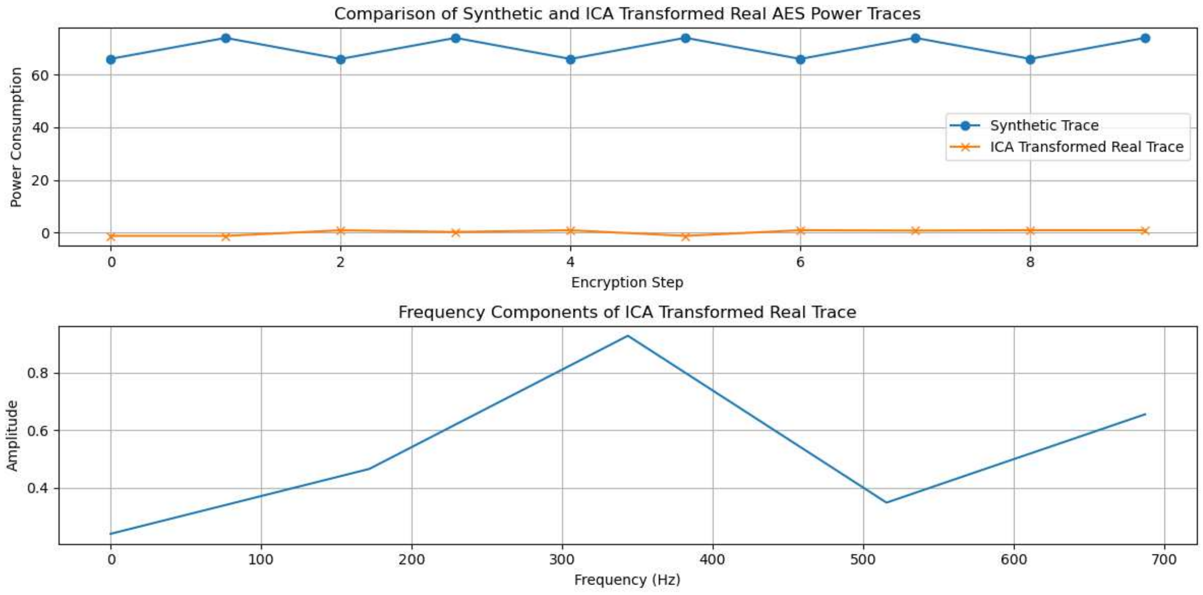


Figure 3.12: Comparison using independent component analysis

indicates a significant degree of similarity between the traces, with a slight time shift required for optimal alignment. The Dynamic time warping analysis, shown in the

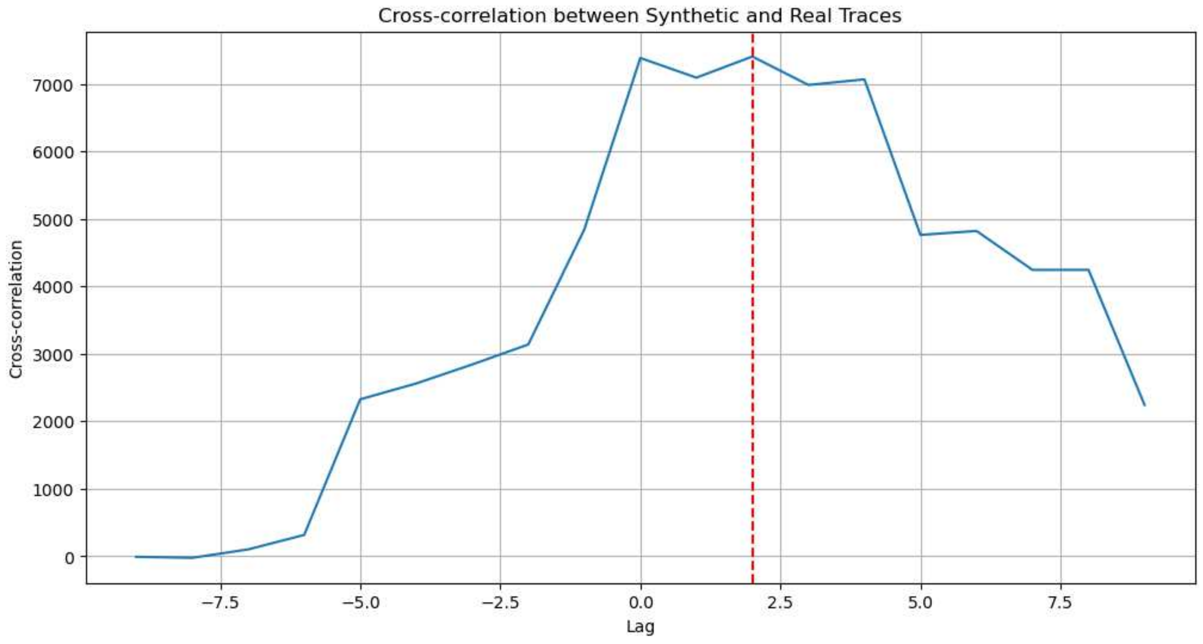


Figure 3.13: Comparison using cross-correlation analysis

Fig. 3.14, measured the alignment between the real and synthetic traces with a DTW distance of 192.935. Post-alignment, the correlation improved to 0.281 as shown in Fig. 3.15 indicating that DTW effectively aligns the traces for better comparison, although some discrepancies remain.

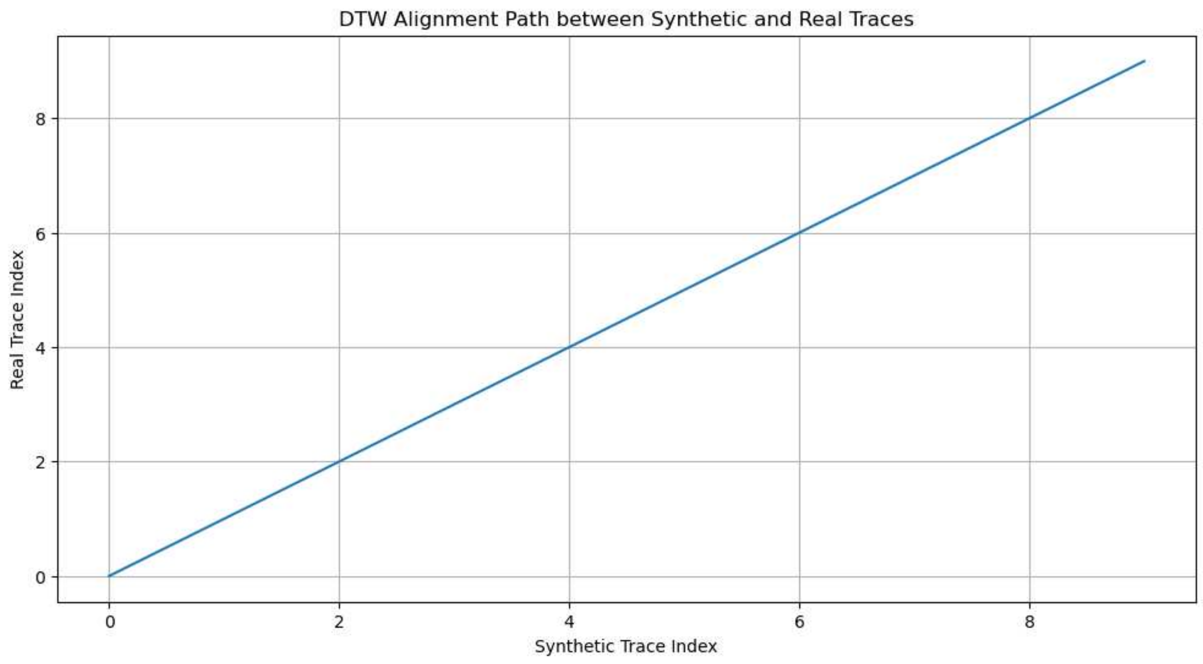


Figure 3.14: DTW Alignment path between synthetic and real traces

The wavelet transform results, depicted in Fig. 3.16 show varying levels of correlation at different decomposition levels. While the correlation for Level 0 and Level 1 coefficients is -1.0, indicating perfect inverse correlation, Levels 2 and 3 show positive correlations of 0.9969 and 0.7555, respectively. The overall correlation between the reconstructed synthetic and real signals is 0.3223, suggesting that the wavelet transform captures the multi-resolution aspects of the traces effectively as shown in Fig. 3.17.

The comparison between actual and predicted real traces using linear regression shows a moderate positive correlation of 0.281 as indicated in Fig. 3.18. This indicates that the predicted traces follow the general trend of the actual traces but with significant deviations. The R-squared score of 0.079 suggests that the linear model explains only 7.9% of the variance in the real power consumption data.

The comparison between actual and predicted real traces using a Random Forest

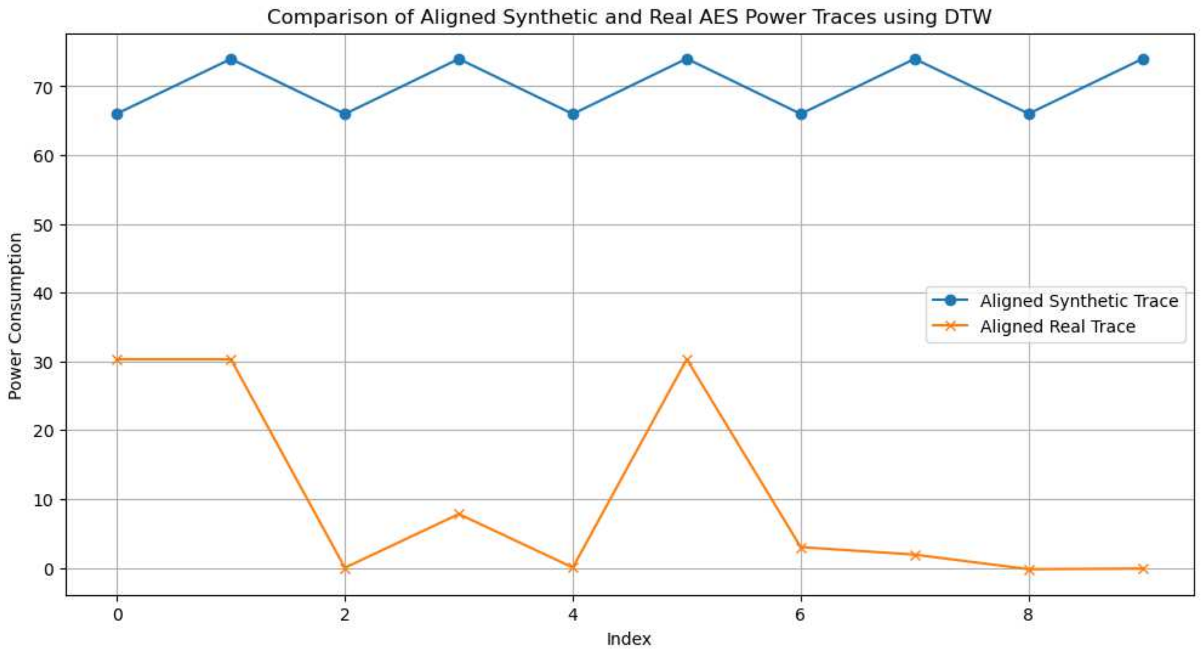


Figure 3.15: Comparison of real and synthetic power trace using DTW

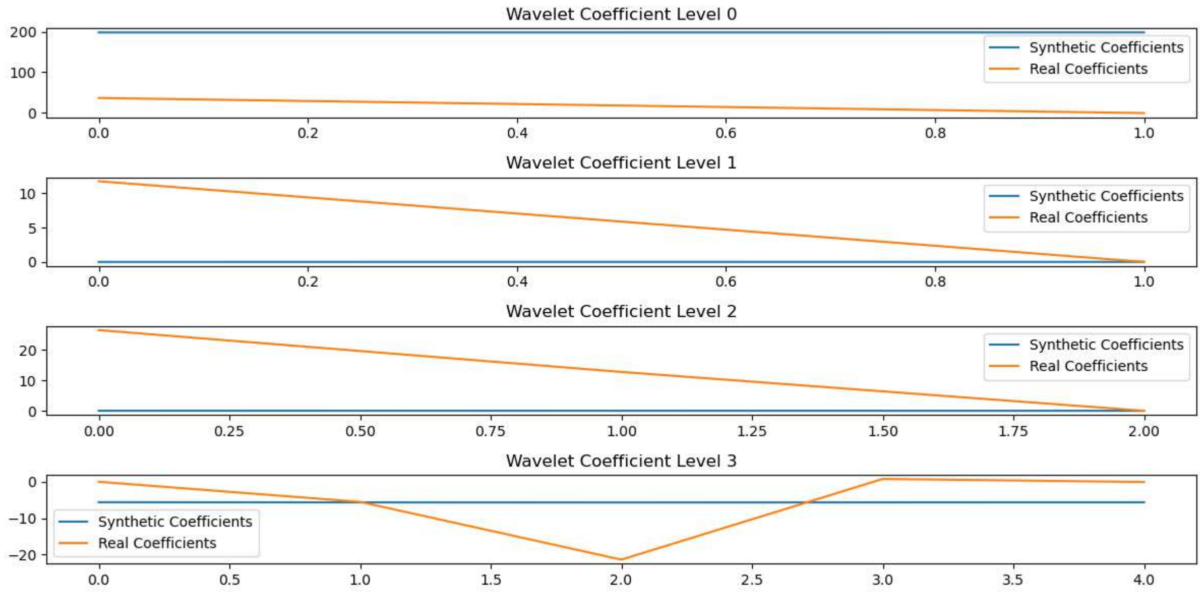


Figure 3.16: Correlation for various wavelength coefficients from level 0 to level 3

model demonstrates a high correlation of 0.986, indicating that the predicted traces closely follow the actual traces as shown in Fig. 3.19. The R-squared score of 0.767 suggests that the Random Forest model explains 76.7% of the variance in the real

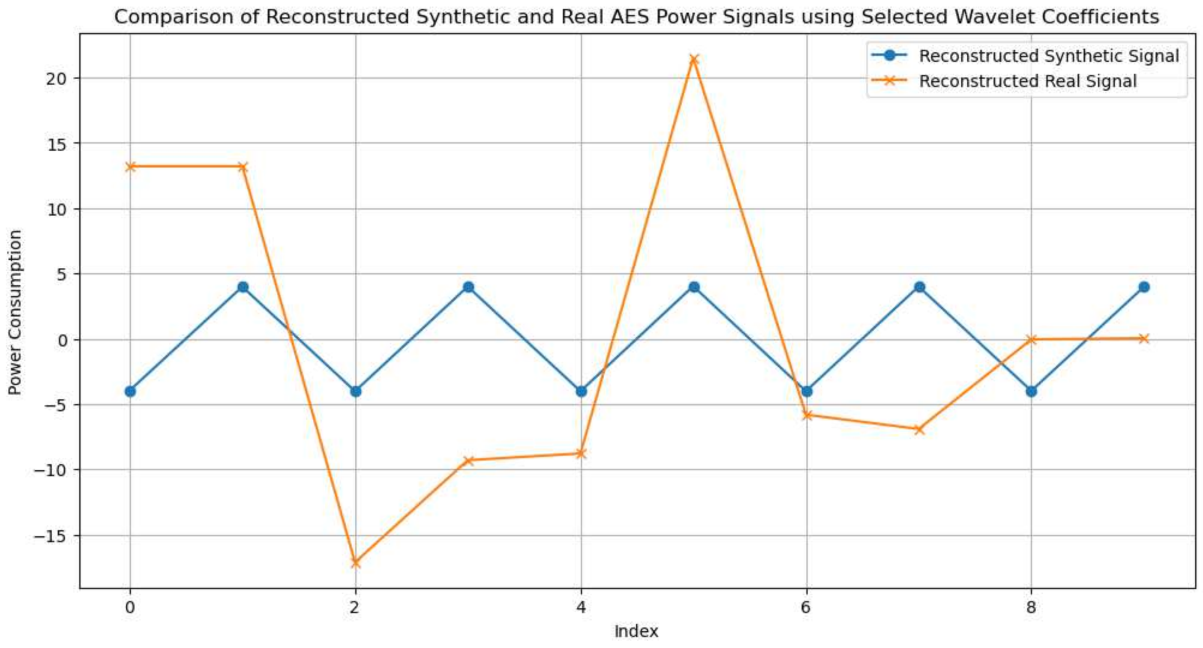


Figure 3.17: Comparison of reconstructed synthetic and real aes power traces using selected wavelength coefficients

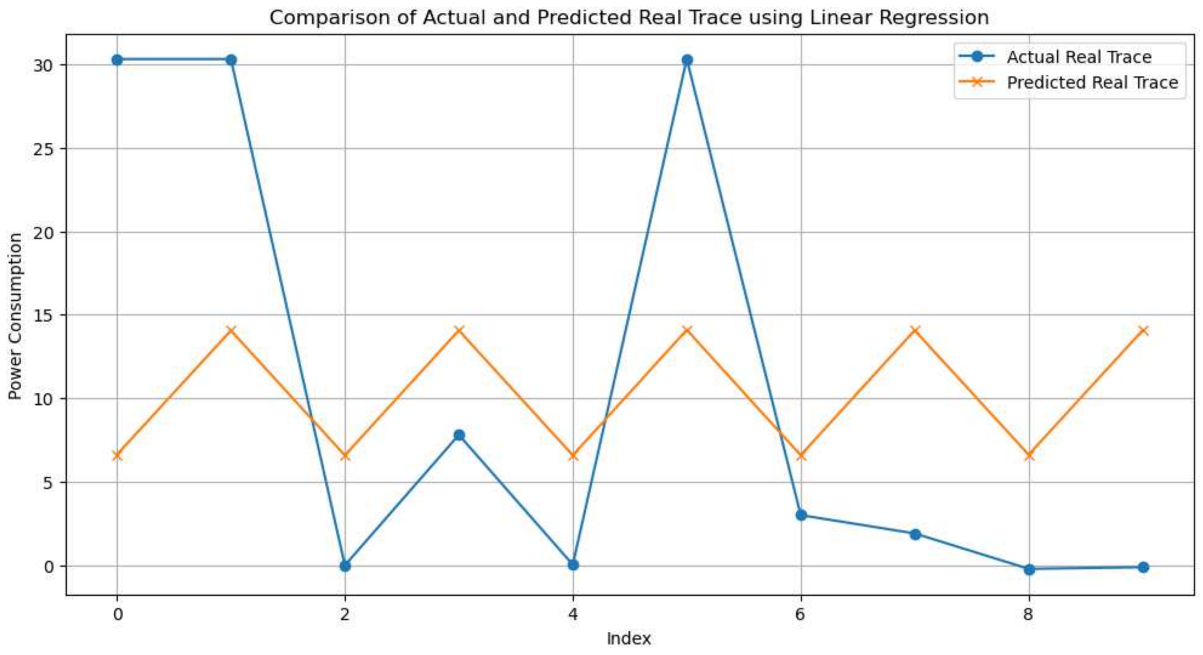


Figure 3.18: Comparison using linear regression

power consumption data. This high R-squared value and correlation indicate that the Random Forest model is effective in capturing the complexities of the power trace data, making it a suitable method for analyzing the relationship between synthetic and real traces.

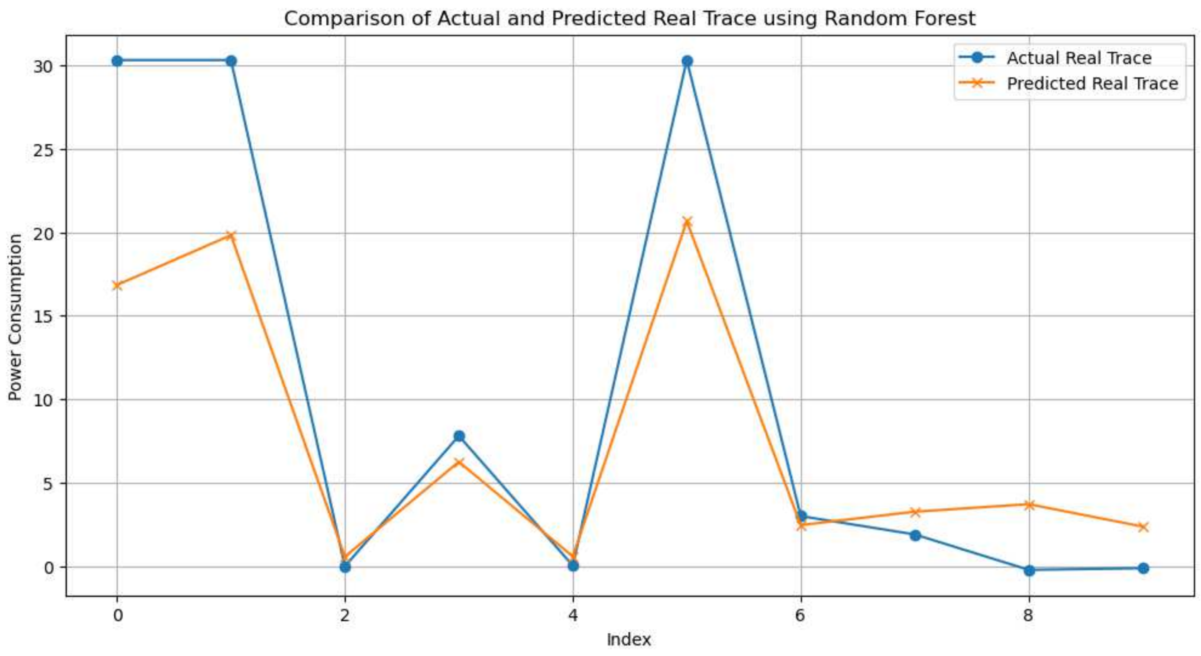


Figure 3.19: Comparison using random forest

3.3 Summary

The first module of our research emphasized on the collection and analysis of power traces for masked AES implementation from both real and synthetic sources. The successful collection of real power traces using the Syscomp CGR-101 oscilloscope provided us an in-depth understanding of how power consumption patterns correlate with cryptographic operations.

In parallel, synthetic power traces were generated using a Python script based on the Hamming Weight model. These traces aimed to mimic the real power traces collected, allowing for a comparative analysis. By applying filtering and smoothing techniques, the quality of synthetic power traces was enhanced, reducing noise and increasing the fidelity of the data.

A detailed timing analysis was conducted to align the real and synthetic traces, exploring the temporal aspects of power consumption patterns. This alignment was crucial in achieving a high level of correlation between the two types of traces. Various techniques, including Fast-Fourier Transform, were employed to compare the traces, with results indicating varying degrees of similarity.

Among the analysis techniques, the Random Forest method showed the most promising results, achieving a correlation coefficient of 0.9857 and an R-squared score of 0.7666. This high degree of similarity between the synthetic and real power traces validates the effectiveness of the synthetic generation method and highlights the potential of using synthetic data to augment real training data in side-channel attacks analysis.

To conclude, this module showed that side-channel attacks analysis could benefit from the addition of artificial power traces to actual training data. The results demonstrate the utility of synthetic data in situations when real power trace collection is difficult or not feasible. This work also prompts us to strengthen the defences of cryptographic systems against side-channel attacks by verifying the validity of synthetic traces.

Chapter 4

Module 2 : Feature extraction, classification and retracing of power traces

Module 2 explores various feature extraction and classification methods to analyze the collected electromagnetic emissions or power traces of different AES implementations. We focus particularly on the AES_PTv2 dataset, which contains power traces from three types of AES implementations: unprotected, Masked1, and Masked2, all executed on the Piñata board. The primary objective of this module is to extract significant features from the dataset, classify the traces according to their corresponding AES implementations, and then investigate the potential of retracing the original power traces from the derived feature set. Our goal is to enhance the understanding of different AES implementations, identify their vulnerabilities, and ultimately improve the security of the AES algorithm. Figure 4.1 provides an overview highlighting Module 1 of the research contributions.

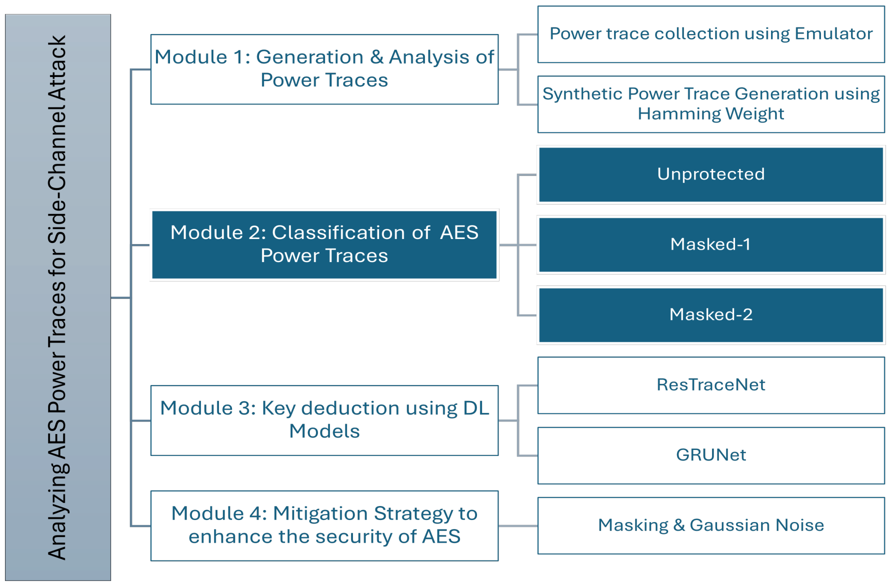

Analyzing AES Power Traces for Side-Channel Attack
Module 1: Generation & Analysis of Power Traces
Power trace collection using Emulator
Synthetic Power Trace Generation using Hamming Weight
Module 2: Classification of AES Power Traces
Unprotected
Masked-1
Masked-2
Module 3: Key deduction using DL Models
ResTraceNet
GRUNet
Module 4: Mitigation Strategy to enhance the security of AES
Masking & Gaussian Noise

Figure 4.1: Research Overview: Highlighted Module 2

4.1 Dataset

The AES_PTv2 is a comprehensive dataset featuring power and EM collected from several devices executing various AES implementations—both unprotected and protected. The primary objective of this dataset is to provide realistic power and EM emissions from real devices, reflecting actual cryptographic operations. This dataset addresses the absence of an open dataset for conducting side-channel analysis using different devices and implementations [97].

4.1.1 Dataset overview

The dataset consists of power traces collected from the Piñata board, an embedded device based on a microcontroller from the ARM Cortex-M4 family. The dataset follows the HDF5 hierarchical structure but has been converted into CSV files for easier usage. All traces are gathered during the encryption process, focusing on three different types of AES software implementations:

- **Unprotected AES Implementation:** A traditional software implementation (AES-128 in ECB mode) without any masking techniques.
- **Masked Scheme 1 (MS1):** A weak implementation that uses a masked lookup table for the SBox operation, wherein the output mask is removed after every 1-byte lookup, resulting in a clear correlation between the mask and the SBox time window.
- **Masked Scheme 2 (MS2):** A robust implementation where the output mask is discarded after the ShiftRows operation, ensuring that the mask isn't leaked during the SBox computation, thus providing a deeper layer of security against attacks.

4.1.2 Piñata Board

The Piñata board, created by Riscure, is based on an ARM Cortex-M4F core operating at a 168 MHz clock speed. It has been modified and programmed to serve as a training target for side-channel analysis (SCA) and fault injection.

4.1.3 Data Organization

The dataset follows a hierarchical structure and is organized into the following subgroups:

- **Based on Device:** Each device has three subgroups corresponding to each AES implementation.
- **Based on Operation:** Each subgroup is further divided into profiling (random keys) and attack (fixed keys) phases.
- **Based on Trace:** Each subgroup contains traces, labels, and metadata.

Fig 4.2 and 4.3 shows the dataset structure based on the device, operation and trace.

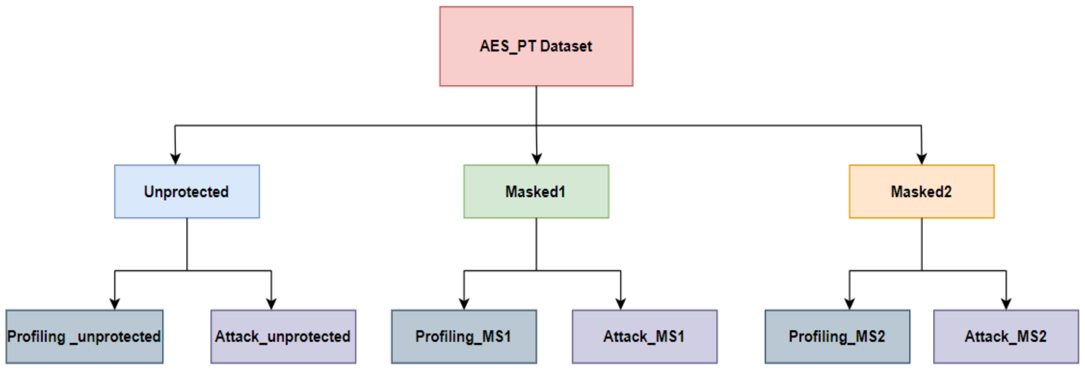


Figure 4.2: AES_PT dataset in hierarchical structure(based on the device and operation)

4.1.4 Number of traces

- **Unprotected AES:** 150,000 power traces (100,000 with random keys and 50,000 with a fixed key).

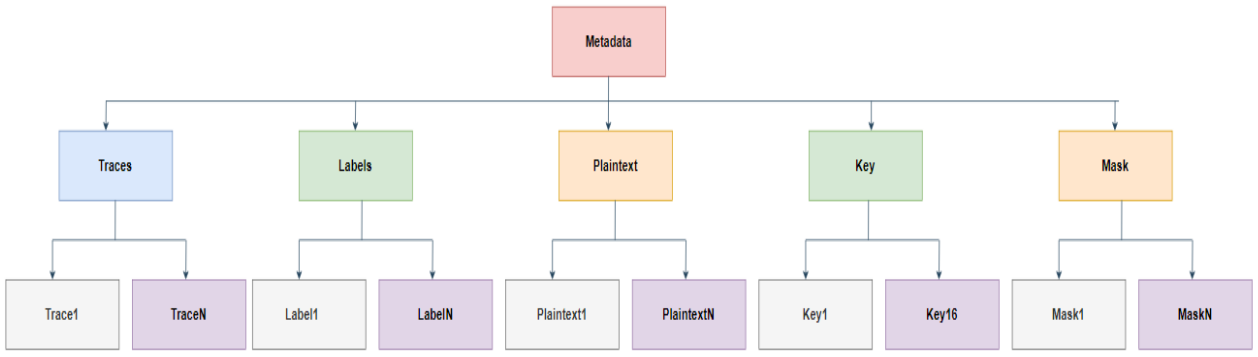


Figure 4.3: AES_PT dataset in hierarchical structure (based on the metadata)

- **MS1:** 200,000 power traces (150,000 with random keys and 50,000 with a fixed key).

- **MS2:** 300,000 power traces (200,000 with random keys and 100,000 with a fixed key).

4.1.5 Trace Collection

- **Measurement Setup:** The board's power consumption during AES encryption is measured using a Tektronix CT1 current probe connected to a 20 GS/s digital oscilloscope (LeCroy Waverunner 9104). The oscilloscope is triggered by a GPIO signal from the microcontroller when internal computation starts.

- **Sample Details:** Each power trace consists of 1,260 samples (1,500 and 1,800 for the masked implementations 1 and 2, respectively) taken at 1 GHz with 8-bit resolution, corresponding to the first SBox operation.

- **Trace Acquisition:** Devices encrypt 16-byte random plaintexts using the three AES implementations. Power consumption is measured with a Langer EM probe attached to the oscilloscope, which is triggered by the microcontroller's GPIO signal. The high-sensitivity probe is placed over a decoupling capacitor on the power line of the device.

- **Preprocessing:** Traces are preprocessed by applying zero mean, standardization, waveform realignment, and a lightweight software low-pass filter.

Applications

The AES PTv2 dataset is widely used in the research community for:

- Developing and Testing Attack Techniques: Researchers use the dataset to develop and validate side-channel attack techniques, such as Differential Power Analysis (DPA) and Correlation Power Analysis (CPA). These techniques aim to exploit the information leaked through power consumption to recover secret keys.

- Evaluating Countermeasures: The dataset is also used to evaluate the effectiveness of countermeasures, such as masking and noise addition, that are designed to protect cryptographic implementations from side-channel attacks.

- Machine Learning and Deep Learning Research: The dataset serves as a benchmark for testing machine learning and deep learning models that are trained to predict cryptographic keys based on power traces. Researchers explore various models, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to improve attack accuracy.

Challenges and Considerations

- High Dimensionality and Noise: The power traces can be high-dimensional and noisy, presenting challenges for analysis. Researchers must preprocess the data, reduce dimensionality, and handle noise to extract meaningful features.

- Realistic Scenarios: The dataset may include traces with intentional desynchronization or added noise to simulate real-world conditions, where measurements may not be perfectly aligned or clean. This adds complexity to the analysis and requires robust methods to handle these challenges.

- Ethical and Security Concerns: While the dataset is valuable for advancing the understanding of side-channel attacks, it also raises ethical and security concerns. Researchers must consider the implications of their work and ensure that it is used to improve security rather than exploit vulnerabilities.

4.2 Methodology

4.2.1 Classification of Power Traces

In this section, feature extraction techniques are applied to the statistical properties of the power traces, and a Support Vector Machine (SVM) classifier is utilized to identify significant features and classify the traces into their respective AES implementations.

Initial Feature Extraction and Classification

Major statistical properties were identified and then used in feature extraction to select the best features that contribute the most to the model's predictive power in classifying the traces. Initially, statistical properties are extracted from the power traces, including mean, standard deviation (std), minimum (min), maximum (max), median, first quartile (Q1), third quartile (Q3), interquartile range (IQR), delta75-25 (difference between the 75th and 25th percentiles), and coefficient of variation (cv). Using these features, an SVM classifier is employed to identify significant features and classify the power traces into two implementations: unprotected and masked1. An SVM classifier is a supervised machine learning model that is optimal for classification tasks, finding the hyperplane that best separates different classes in the available feature set [98].

A short description of each statistical property used is provided below:

- **Mean:** The average value of the dataset, providing a central value around which the data points are distributed [99].
- **Standard deviation:** A measure of the amount of variation or dispersion in the dataset, indicating how spread out the data points are from the mean [99].
- **Minimum (min):** The smallest value in the dataset, helping to identify the lower bound of the data range [99].

- **Maximum (max):** The largest value in the dataset, helping to identify the upper bound of the data range [99].
- **Median:** The middle value of the dataset when sorted in ascending order, provides a measure of central tendency that is less affected by outliers compared to the mean [99].
- **First Quartile (Q1):** The value below which 25% of the data points fall, helping to understand the lower part of the data distribution [99].
- **Third Quartile (Q3):** The value below which 75% of the data points fall, helping to understand the upper part of the data distribution [99].
- **Interquartile Range (IQR):**The range between the first quartile (Q1) and the third quartile (Q3), measuring the spread of the middle 50% of the data points, providing insight into the variability within the central portion of the dataset [99].
- **Delta75-25:** The difference between the 75th percentile and the 25th percentile, another measure of spread [99].
- **Coefficient of Variation (cv):** The ratio of the standard deviation to the mean, provides a normalized measure of dispersion [99].

The focus is on the multiple feature extraction techniques used to analyze the collected power traces and EM emissions from the AES_PTv2 dataset. The aim is to recognize and identify which features contribute the most to the predictive power of the model, thereby enhancing the classification process and analysis of the various AES implementations, namely unprotected and masked1, using a support vector machine (SVM) classifier. The techniques used for feature extraction are listed below:

- **Permutation importance:** This method analyses the importance of each feature by computing the decrease in the performance of the model when the values are shuffled randomly. It aids us in understanding the contribution of each feature towards making predictions [100].

- **Recursive feature elimination:** This technique as the name suggests recursively removes the least important feature and builds the model with the remaining features helping us identify features that are integral for the performance of the model thus improving accuracy [100].

- **Mutual information:** This method measures the dependency between every feature and the target variable ie the amount of information one variable holds about another variable. It quantifies the quantity of knowledge that is obtained about the output variable with the help of each feature, thus finding features that has the most predictive power [101].

- **Principal component analysis:** This is a dimensionality reduction technique that converts the original set of variables into a smaller set of variables called principal component and these components would have the highest variance ,thus enabling in simplification of the dataset [101].

- **L1 Regularization:** Lasso (Least Absolute Shrinkage and Selection Operator) is a regression technique that can do both selection of features and regularization. It adds a penalty value that is equal to the absolute value of the magnitude of coefficients. If the coefficient value I closer to zero, the feature selection performed would be precise and lead to a simple model [101].

- **SHapley Additive exPlanations (SHAP):** SHAP works on the concept of cooperative game theory and provides a unified measure of feature importance. It explores the impact each feature has on the output of the model by taking into account the contribution of every feature towards the prediction thus offering a detailed approach to feature selection [101].

4.2.2 Focus on PCA for improved classification

To further improve classification accuracy, Principal Component Analysis (PCA) is employed to identify the most important features from the set of extracted statistical properties. PCA reduces the dimensionality of the data while retaining the most significant variance-contributing features, streamlining the classification process. After determining the optimal features using PCA, a Support Vector Machine (SVM) classifier is applied to classify the power traces into the respective AES implementations: unprotected, masked1, and masked2. The SVM classifier leverages the principal components identified by PCA to make accurate classifications. To validate the classification results, a confusion matrix, classification report, and distribution of features are utilized. These metrics provide a comprehensive assessment of model performance, highlighting the accuracy and reliability of the classification process based on the features extracted using PCA. The confusion matrix shows the true positives, false positives, true negatives, and false negatives, while the classification report includes precision, recall, and F1 scores for each class. The distribution of features offers insights into how the features are spread across different classes, ensuring that the most relevant features effectively contribute to the classification of power traces into their corresponding AES implementations. As shown in Algorithm 2, the PCA-based feature selection process involves computing statistical properties, performing PCA, and selecting top features based on their contribution ranks.

4.2.3 Optimal Threshold Calculation and Retracing the Power Traces

After classifying the power traces into their respective AES implementations, the next crucial step involves calculating the optimal threshold and retracing the original power traces from the derived dataset, which includes selected statistical properties. This process employs techniques such as hashing and metadata analysis. Fig 4.4 shows the feature extraction, classification, and optimal threshold calculation.

Threshold Calculation

Identifying the optimal threshold during classification is critical for achieving accurate and reliable outputs. The optimal threshold is the point that best segregates the various classes (unprotected, masked1, and masked2), indicating the classifier's performance. This involves computing the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) for each class. The optimal threshold calculation process involves:

Algorithm 2 PCA-Based Feature Selection for choosing optimal statistical properties

```
Input: TD: Dataset with traces from Unprotected, Masked1, and Masked2 implementations, labeled 0, 1, and 2 respectively
SP: Array of statistical properties [Mean, Std, Min, Max, Median, Q1, Q3, IQR, Skewness, Kurtosis, Delta 75-25, CV]
n_comp: Number of principal components to choose (default:10)
Best_features: Best features collected using PCA
Output: Best_Features: List of selected features based on PCA
9  Function Compute_Statistical_Properties(TD, SP):
10     return Computes the specified statistical properties for each sample and returns a new dataset with the computed properties
11 Function PCA_contribution(Input_Features):
12     Performs PCA on the input features and returns a ranked list of features based on their contribution to the principal components
13 Function Select_Top_Features(Rank_Features):
14     Selects the top features based on their contribution ranks
15 Algorithm PCA_Based_Feature_Selection(TD, SP, n_comp):
16     Current_Dataset ← TD  Current_Features ← SP  X_PCA ← []  Best_Features ← []
       // Compute Statistical Properties
17     Current_Dataset ← Compute_Statistical_Properties(Current_Dataset, SP)  Reduced_Features ← [Mean, Std, Min, Max, Median, Q1, Q3, Skewness, IQR, Kurtosis]  X_PCA ← PCA(Current_Dataset[Reduced_Features], n_comp=n_comp)
18     for component in range(n_comp) do
19         Rank_Features ← PCA_contribution(X_PCA[component])  Best_Features ← Select_Top_Features(Rank_Features)
20     end
21     return Best_Features
22 return
```

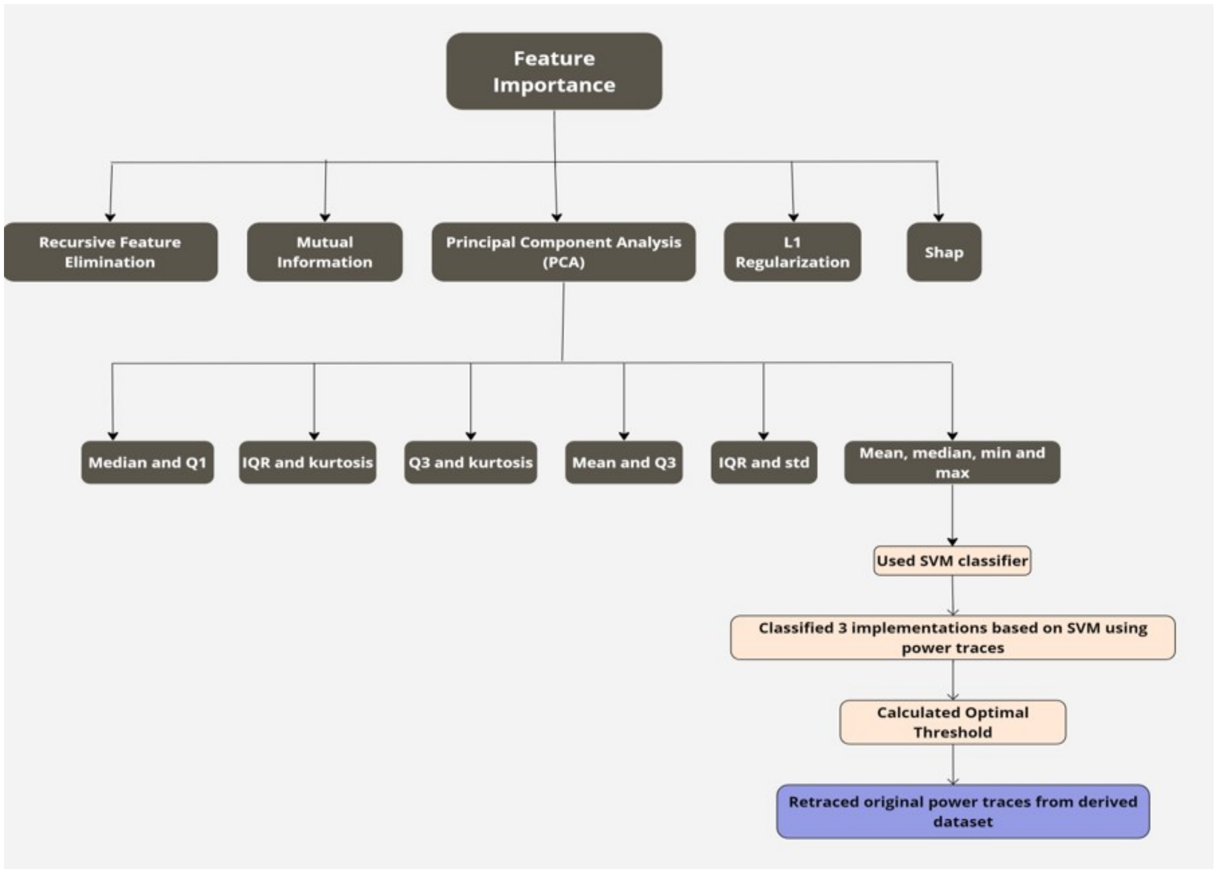

Feature Importance
Recursive Feature Elimination
Mutual Information
Principal Component Analysis (PCA)
L1 Regularization
Shap
Median and Q1
IQR and kurtosis
Q3 and kurtosis
Mean and Q3
IQR and std
Mean, median, min and max
Used SVM classifier
Classified 3 implementations based on SVM using power traces
Calculated Optimal Threshold
Retraced original power traces from derived dataset

Figure 4.4: Feature extraction, classification and optimal threshold calculation

- **Generating the ROC Curve and AUC:** For each class, the ROC curve is plotted, and the AUC is calculated, serving as a single metric to evaluate the model's performance.

- **Determining the Optimal Threshold:** The optimal threshold is found by identifying the point on the ROC curve that maximizes the difference between the True Positive Rate (TPR) and the False Positive Rate (FPR). This point, known as Youden's index, is where the value of TPR - FPR is the highest.

- **Storing Threshold Values:** The calculated threshold values for each class are stored for future classification tasks.

The optimal threshold helps the classifier make accurate decisions by setting the best decision boundary for each class. As shown in Algorithm 3, the process of classification and obtaining optimal threshold and AUC involves computing statistical properties, training an SVM classifier, computing decision scores, and finding optimal thresholds based on the ROC curve.

Retracing the Power Traces from the Derived Dataset Using Hashing and ID Methods

Both the hashing and ID methods require extensive preprocessing, feature extraction, and classification to accurately classify and retrace the power traces. While the hashing technique offers greater security and traceability, the ID method provides a simpler yet effective approach to manage data points and tracing them back to the original raw power traces.

Hashing Method

This method involves generating a unique identifier for each row in the dataset by applying a cryptographic hash function. This enhances data integrity and traceability throughout the data transformation process. The create_hash function converts each row of the dataset into a JSON string, then applies the SHA-256 hash function to create a unique ID for that row. This ensures the identifier is unique, which is crucial for data retracing and integrity. Performance is evaluated using confusion matrices and classification reports.

ID Method

The ID method assigns a unique identifier to each row based on its corresponding index in the data frame. This approach, while simpler than the hashing method, provides a straightforward way to track data points throughout the entire process. Confusion matrices and classification reports are also generated to assess classifier performance.

Both methods play a pivotal role in ensuring the accuracy and integrity of the classification process, facilitating the reliable retracing of power traces to their original form. As shown in Algorithm 4, the process of retrieving the original dataset from the derived dataset involves creating unique hash values for each row and using these hashes to match and retrieve the original traces.

Algorithm 3 Classification and obtaining optimal threshold and AUC

```
Input: Dataset: Power traces for Unprotected AES (label 0), Masked1 AES (label
    1), and Masked2 AES (label 2)
X_train and X_test: The training data features and The test data features
Y_train: The training data labels
Decision_score: Decision score from classifier
Fpr, Tpr: False positive rate and True positive rate
Threshold: Values used to convert decision score into 0 or 1
Output: Optimal_Thresholds: Dictionary containing optimal thresholds
AUC_Scores: Dictionary containing AUC scores for each class
Function Compute_Statistical_Properties(Dataset):
    Computes Mean, Std, Min, Max for each trace and returns a new dataset
Function Train_SVM(X_train, Y_train):
    Trains an SVM classifier on the given training data
Function Compute_Decision_Scores(Classifier, X_test):
    Computes the decision scores and returns decision scores
Function Compute_ROC_AUC(y_true, decision_scores):
    Computes the ROC curve and AUC score for true labels and decision scores
Function Find_Optimal_Threshold(fpr, tpr, thresholds):
    Finds the optimal threshold and returns the optimal threshold
Algorithm Classification_and_Optimal_Threshold_AUC(Dataset):
    Optimal_Thresholds ← {}
    AUC_Scores ← {}
    Dataset ← Compute_Statistical_Properties(Dataset)
    classifier ← Train_SVM(X_train, Y_train)
    decision_scores ← Compute_Decision_Scores(classifier, X_test)
    for class_index in range(decision_scores.shape[1]) do
        fpr, tpr, thresholds, auc_score ← Compute_ROC_AUC(y_test == class_index, de-
          cision_scores[:, class_index])
        optimal_threshold ← Find_Optimal_Threshold(fpr, tpr, thresholds)
        Optimal_Thresholds[class_index] ← optimal_threshold
        AUC_Scores[class_index] ← auc_score
    end
    return Optimal_Thresholds, AUC_Scores
return
```

Algorithm 4 Retrieving the original dataset from the derived dataset using Hashing

```
Input: Derived_Dataset: Derived dataset
Original_Datasets: List of original datasets (unprotected_df, ms1_df, masked2_df)
SVM_Classifier: Trained SVM classifier on the derived dataset
Output: Org_traces: Original traces retrieved from the original datasets
47 Function create_hash(row):
48     Hashing function to create a unique hash value
49 Algorithm Retrieve_original(Derived_Dataset, Original_Datasets, SVM_Classifier):
50     for label in [0, 1, 2] do
51         unprotected_df['hash_id'] ← unprotected_df.apply(create_hash, axis=1)
52         ms1_df['hash_id'] ← ms1_df.apply(create_hash, axis=1)
53         masked2_df['hash_id'] ← masked2_df.apply(create_hash, axis=1)
54         original_traces ← original_df[original_df['hash_id'].isin(test_df[y_pred == label]['hash_id'])]
55         Org_traces ← pd.concat([Org_traces, original_traces], ignore_index=True)
56     end
57     return Org_traces
58 return
```

4.3 Results and discussion

In this section, we present the results after applying feature extraction techniques to the statistical properties of the power traces. We then evaluate the performance of the classifier using a Support Vector Machine (SVM) and calculate the optimal threshold for each AES implementation.

Feature Extraction and Importance

Several feature extraction techniques were used to identify the most important statistical properties. The techniques employed include permutation importance, recursive feature elimination, L1 regularization, mutual information, and SHapley Additive exPlanations (SHAP). The figures 4.5,4.64.7,4.8,4.9 and 4.10 depict these various feature extraction techniques: permutation importance, recursive feature elimination (RFE), L1 regularization, mutual Information, SHAP, principal component analysis (PCA). The confusion matrix, shown in the 4.11, remains consistent across all feature extraction techniques, highlighting the classification performance of the SVM classifier. The findings from the visualizations are listed below:

- Permutation Importance: Median and Q1 were found to be the most important features.

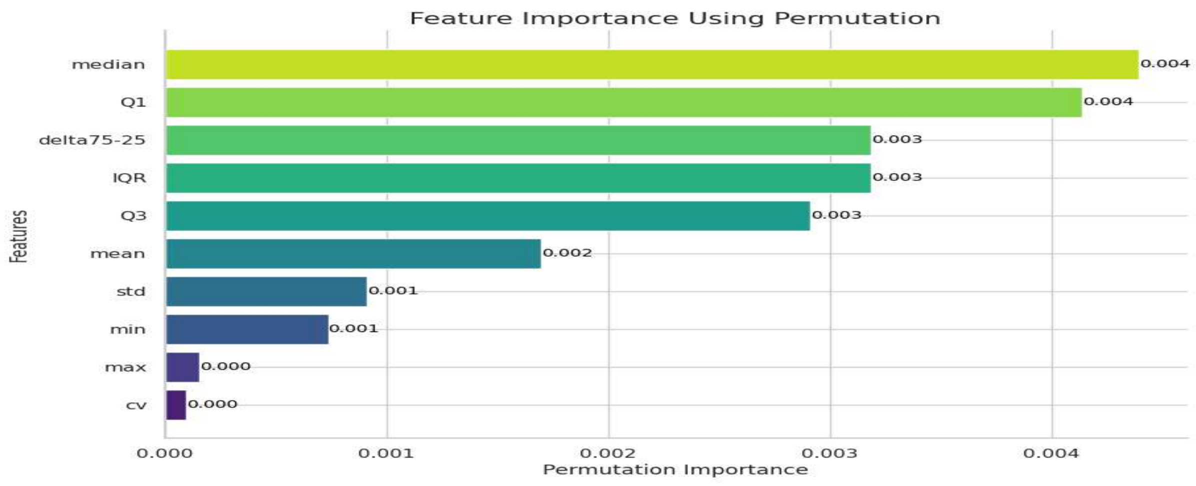


Figure 4.5: Feature extraction using permutation importance

- Recursive Feature Elimination (RFE): The optimal set of features was identified through iterative removal and model evaluation.

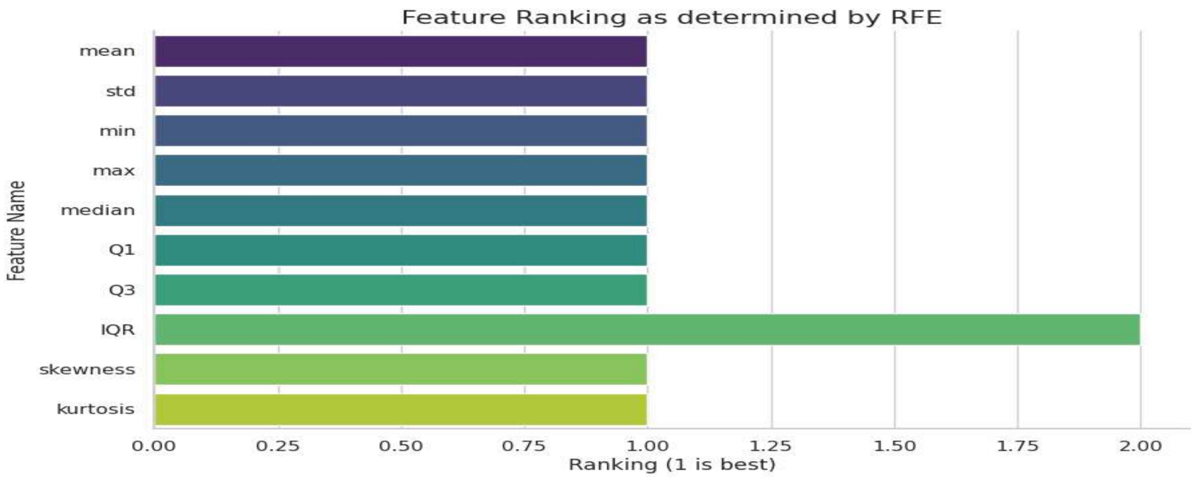


Figure 4.6: Feature extraction using recursive feature elimination

- L1 Regularization: Features such as min, mean, and cv were highly influential.

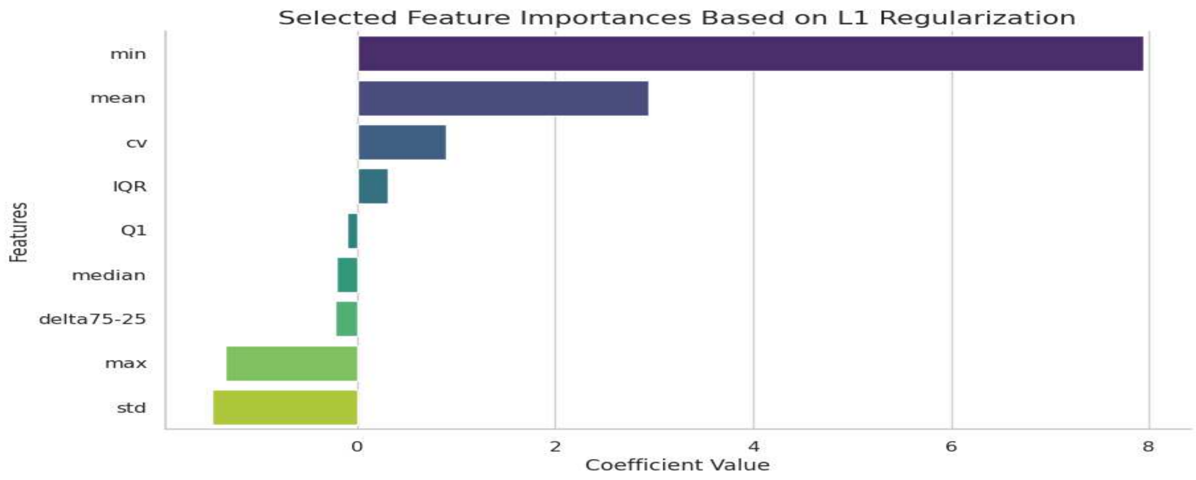


Figure 4.7: Feature extraction using L1

- PCA: Feature combinations such as mean,median,min, max are selected.
- Mutual Information: Key features like mean, min, max, median, and Q1 showed significant mutual information scores.
- SHapley Additive exPlanations (SHAP): Features such as mean, min, and max had a notable impact on the model output.

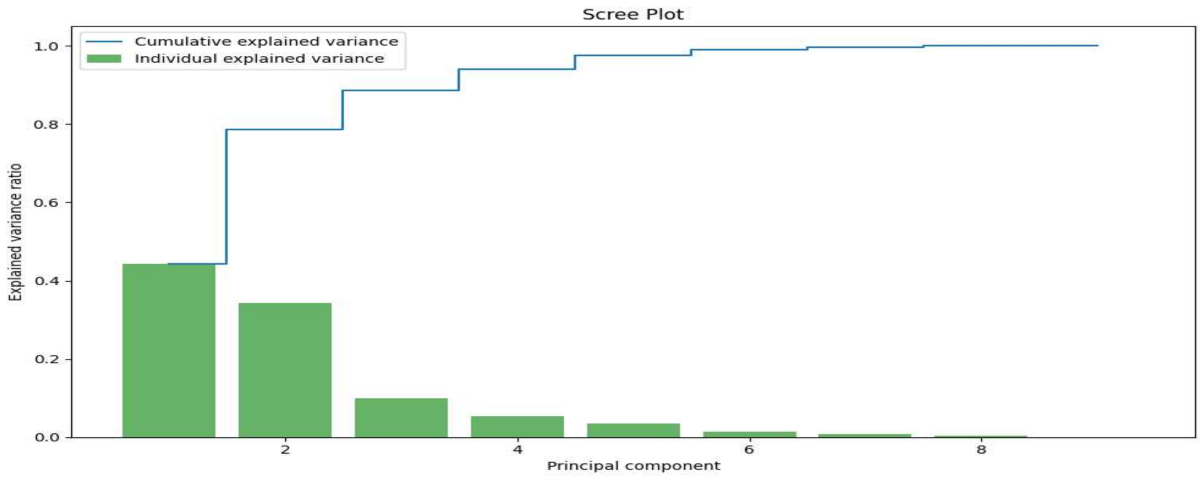


Figure 4.8: Feature extraction using principal component analysis

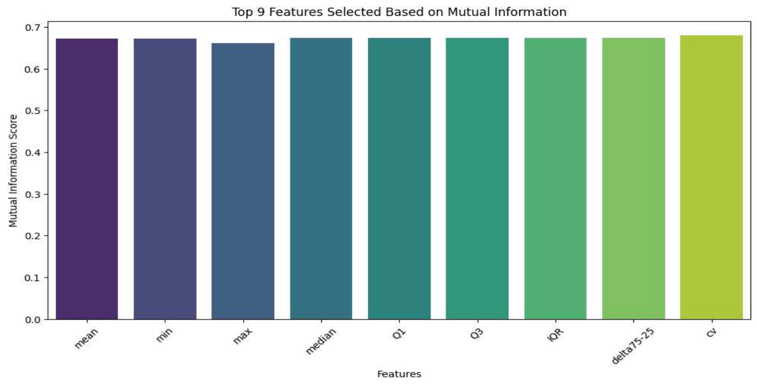


Figure 4.9: Feature extraction using mutual information

PCA and SVM Classification Results

After applying PCA, we identified the following combinations of features: median and Q1, IQR and kurtosis, Q3 and skewness, mean and Q3, IQR and standard deviation, mean, standard deviation, minimum and maximum, and mean, median, minimum, and maximum. We then applied the SVM classifier and tested these features using the attack traces for unprotected and masked AES implementations.

Fig. 4.12- 4.17 shows the confusion matrix for the obtained feature combinations after PCA. Additionally, we calculated the feature distribution by predicted labels, which are depicted in the corresponding figures.

Fig. 4.19 - 4.25 shows the feature distribution graphs. **Key observations from**

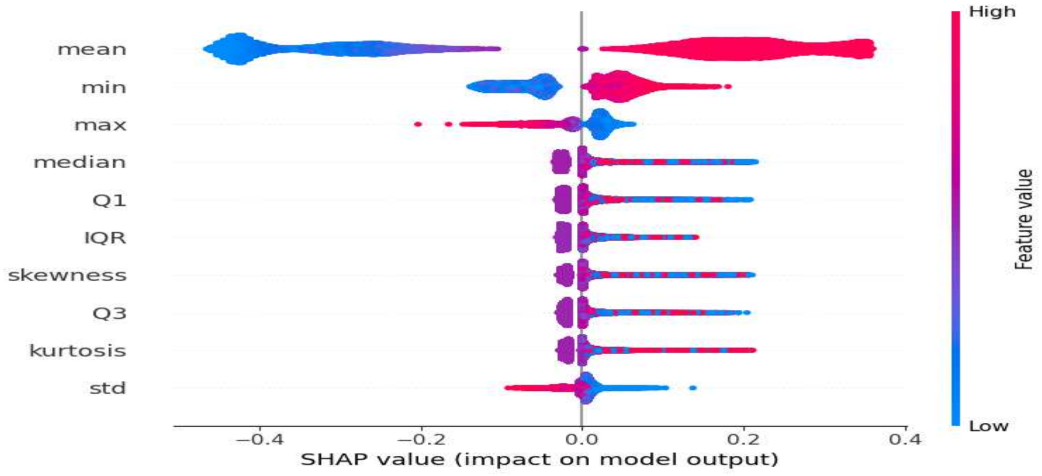


Figure 4.10: Feature extraction using SHAP

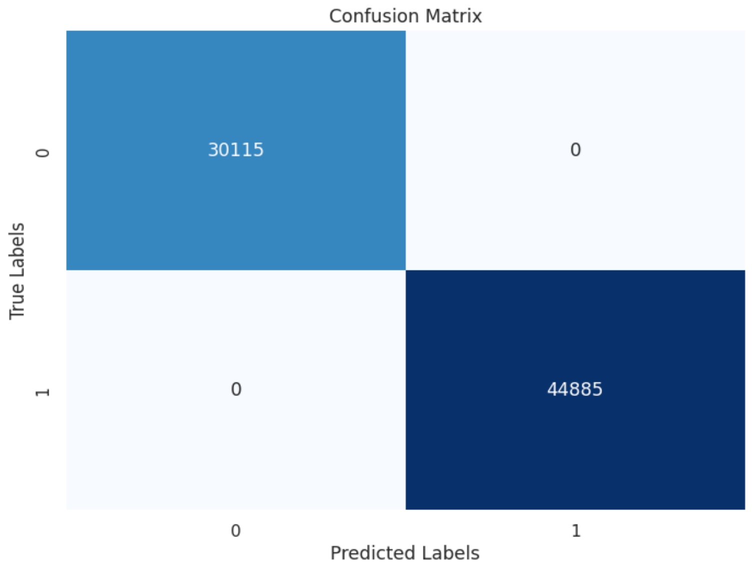


Figure 4.11: Confusion matrix of SVM for all the feature extraction techniques

the feature distribution graphs

Median vs Q1 plot

The orange dots appear to be clustered, indicating that the median and Q1 values of most anticipated Label 1 traces are comparable. Label 0 is represented by blue dots that are also closely clustered. However, some misclassifications or regions where the characteristics do not significantly separate the classes are indicated by a few blue points inside the orange cloud. This is shown in Fig. 4.19.

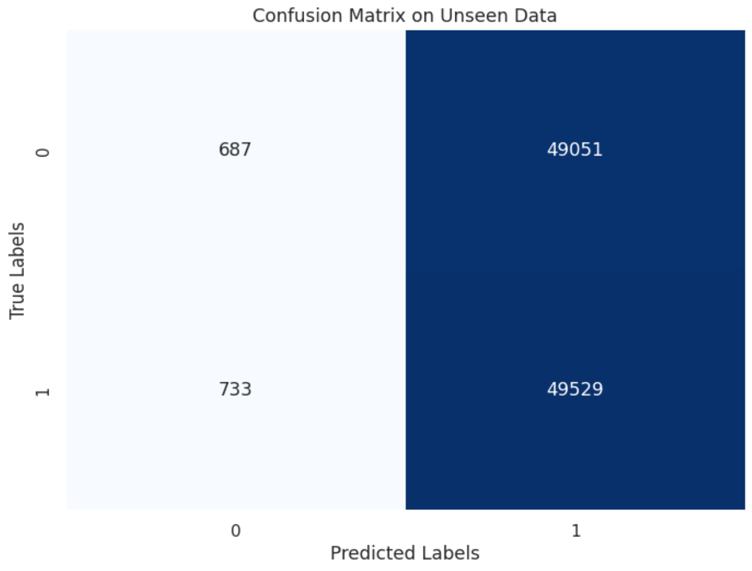


Figure 4.12: Confusion matrix of the model with statistical properties Median and Q1

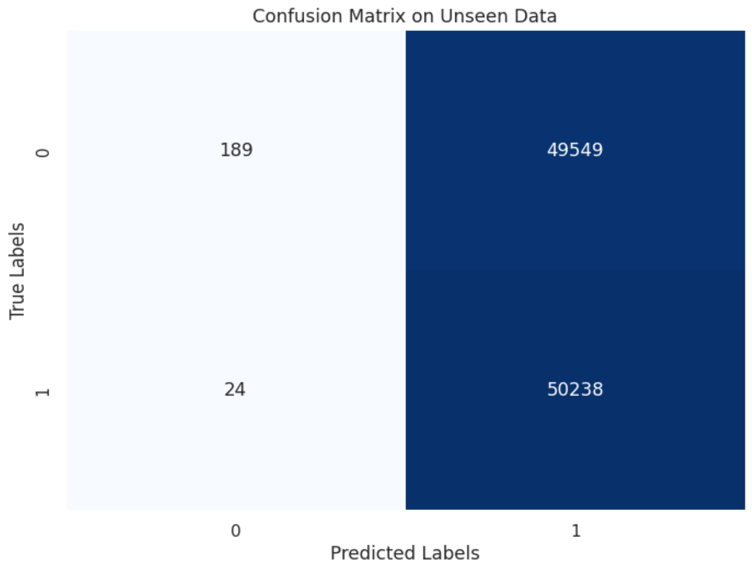


Figure 4.13: Confusion matrix of the model with statistical properties IQR and kurtosis

IQR vs Kurtosis plot

The tightly clustered orange dots in projected Label 1 indicate a high concentration of identical values for IQR and Kurtosis. The blue dots indicate some overlap and difficulties differentiating the groups based alone on these two traits. They also exhibit a larger spread in the IQR values but are still quite close in terms of kurtosis.

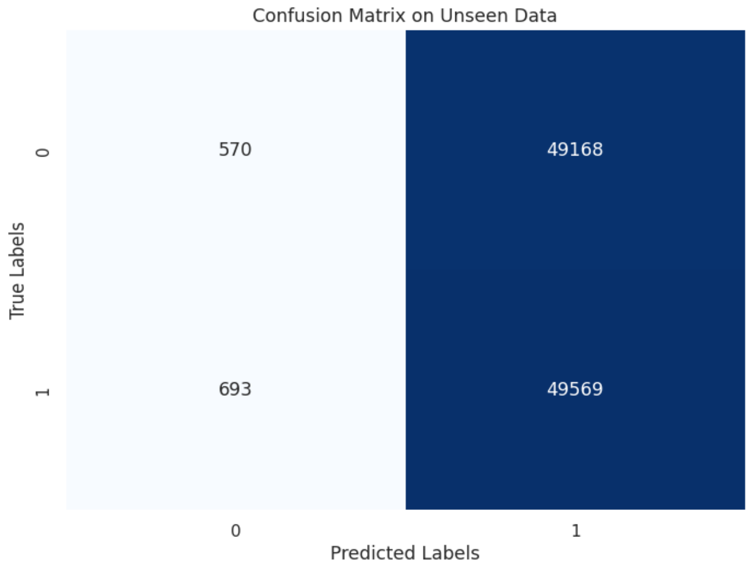


Figure 4.14: Confusion matrix of the model with statistical properties for Q3 and skewness

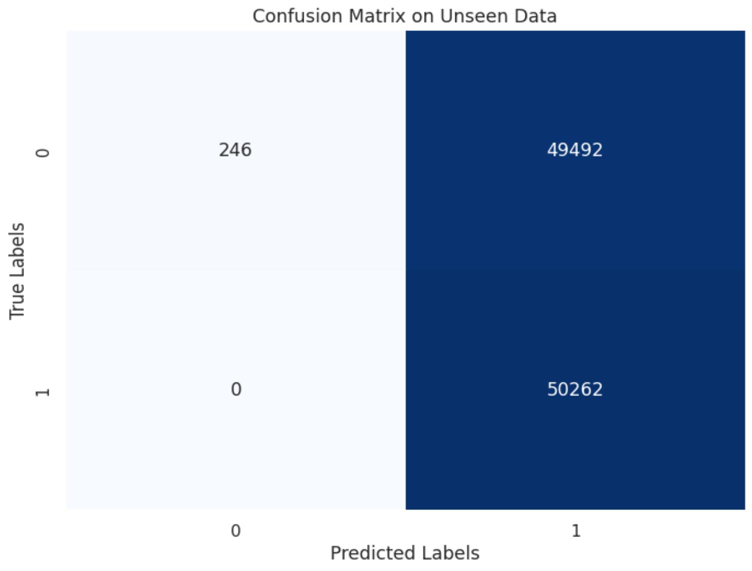


Figure 4.15: Confusion matrix of the model with statistical properties for mean and Q3

Q3 vs Skewness plot

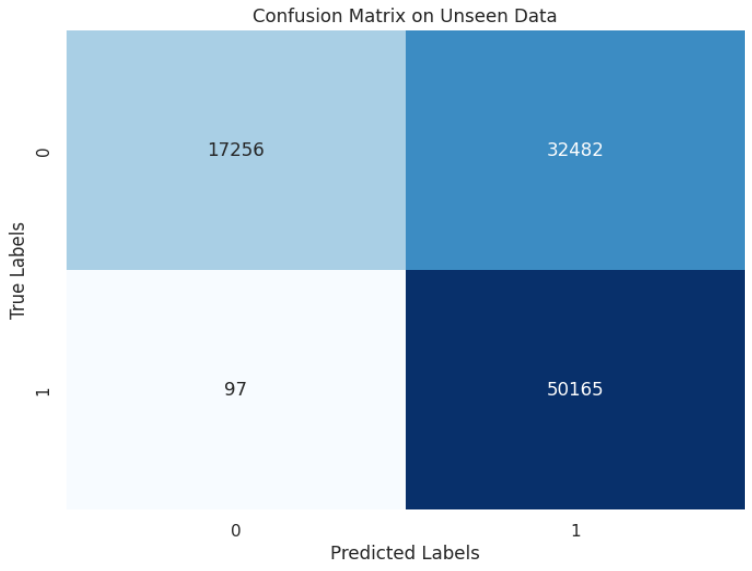


Figure 4.16: Confusion matrix of the model with statistical properties for IQR and std

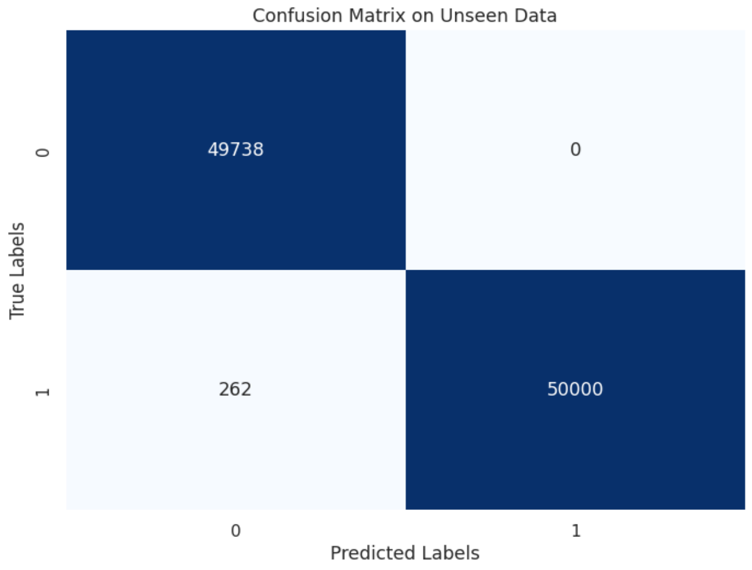


Figure 4.17: Confusion matrix of the model with statistical properties for mean,min,max and std

The scatter plot displays data categorized into two groups based on ”Predicted Label” (0 or 1) across two variables: ”Q3” on the x-axis and ”Skewness” on the y-axis. The plot primarily shows a dense concentration of data points (label 0) with a narrow range of ”Skewness” values and a broad range of ”Q3” values. There is a

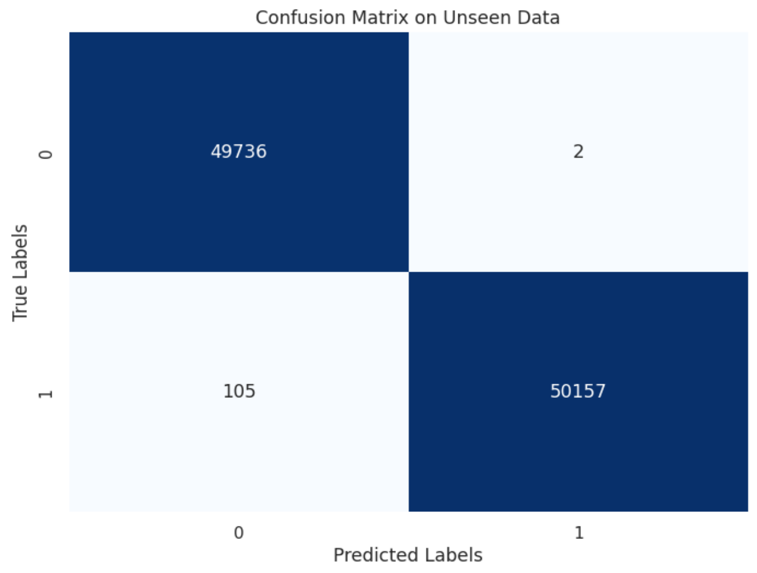


Figure 4.18: Confusion matrix of the model with statistical properties for mean,median,min and max

notable outlier marked as label 1, surrounded by the dense cluster of label 0 points, suggesting a potential anomaly or exceptional case within the dataset.

Mean vs Q3 plot

Label 0 points overlaying label 1 indicates that there could not be a significant difference between the two classes based just on the mean and Q3. If the model predominantly uses these traits, then successfully identifying new samples may become difficult. The dataset's variability is indicated by the dense clustering and wide distribution of the data points. This variability might be advantageous for model training, but it may also call for more intricate modeling strategies or extra features to get acceptable classification results.

IQR vs STD plot

This plot shows a dense cluster where the majority of data points (label 1) show a wide range of skewness across different values of Q3, mainly between -10 and 0. A few anomalies are dispersed. The sparse blue spots (label 0) indicate that there is little difference between the classes.

STD vs Mean

There is a noticeable difference between the two clusters: label 0 represents a cluster with a relatively low mean and standard deviation, whereas label 1 represents a cluster with a wider range of means and greater standard deviations. This suggests that the two anticipated labels may be distinguished from one another using the mean and standard deviation attributes.

STD vs Mean

The same result is obtained as above.

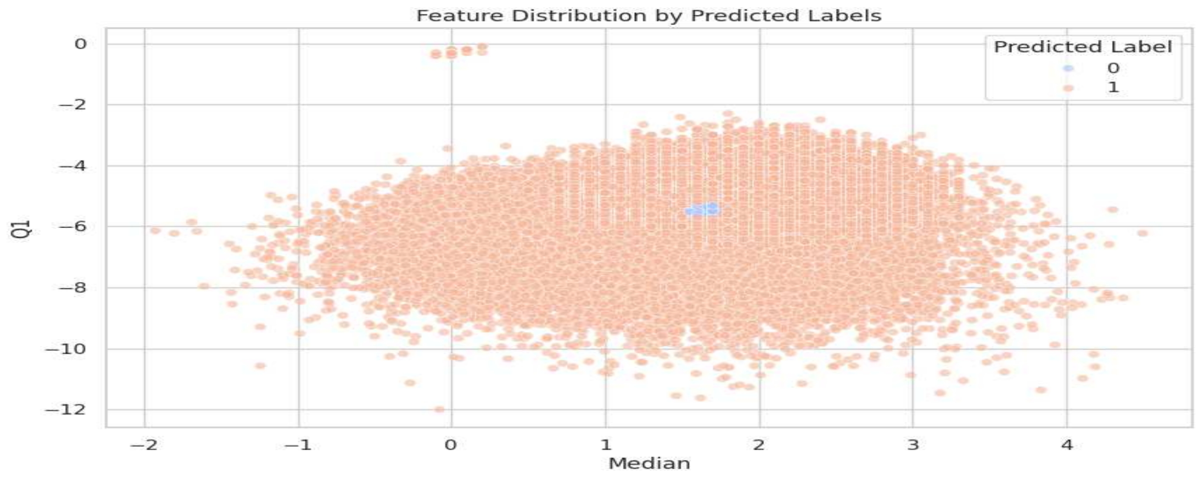


Figure 4.19: Feature distribution graph for statistical properties median and q1

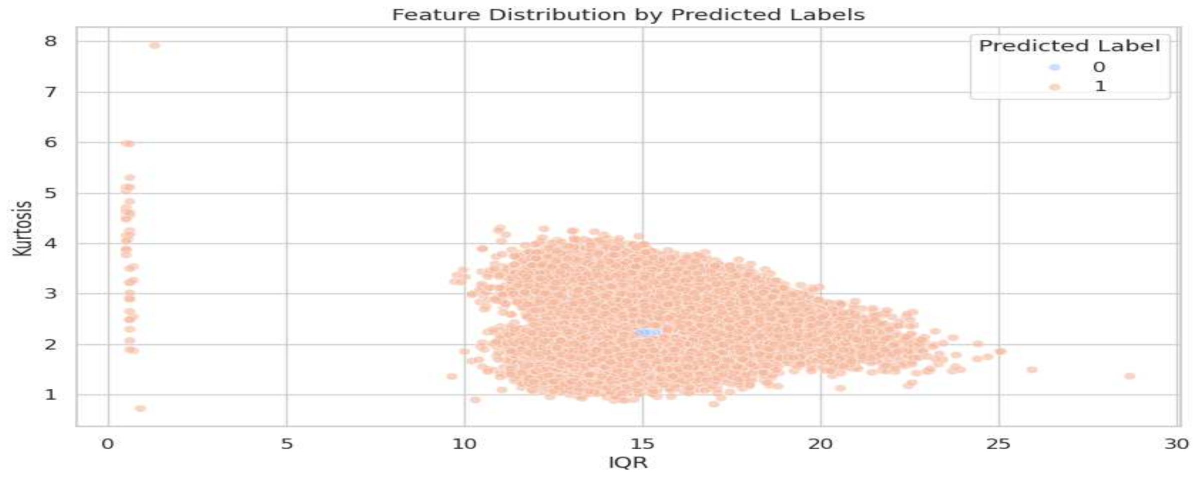


Figure 4.20: Feature distribution graph for statistical properties IQR and kurtosis

As the confusion matrix and feature distribution graphs suggest, the best accuracy

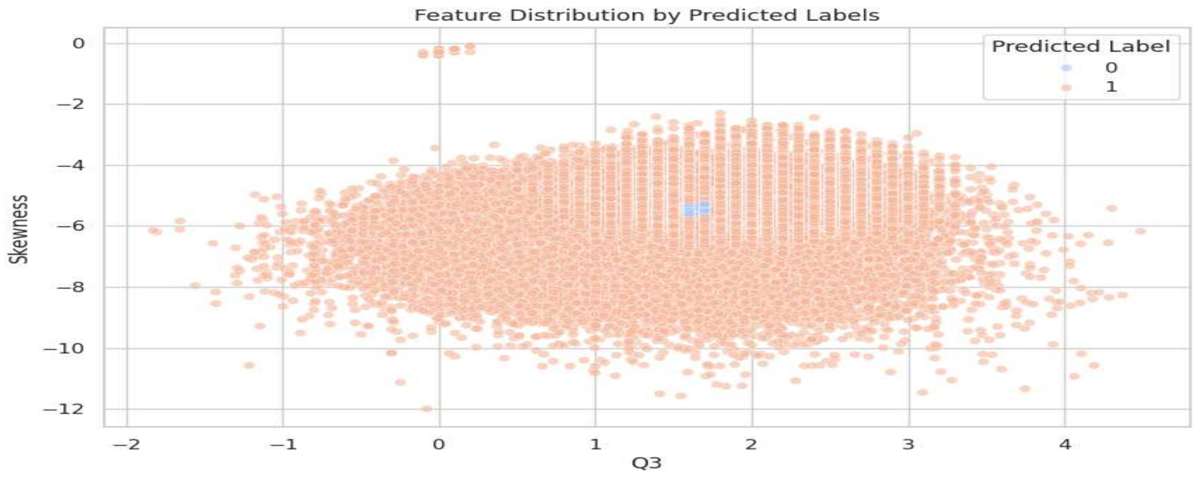


Figure 4.21: Feature distribution graph for statistical properties Q3 and skewness

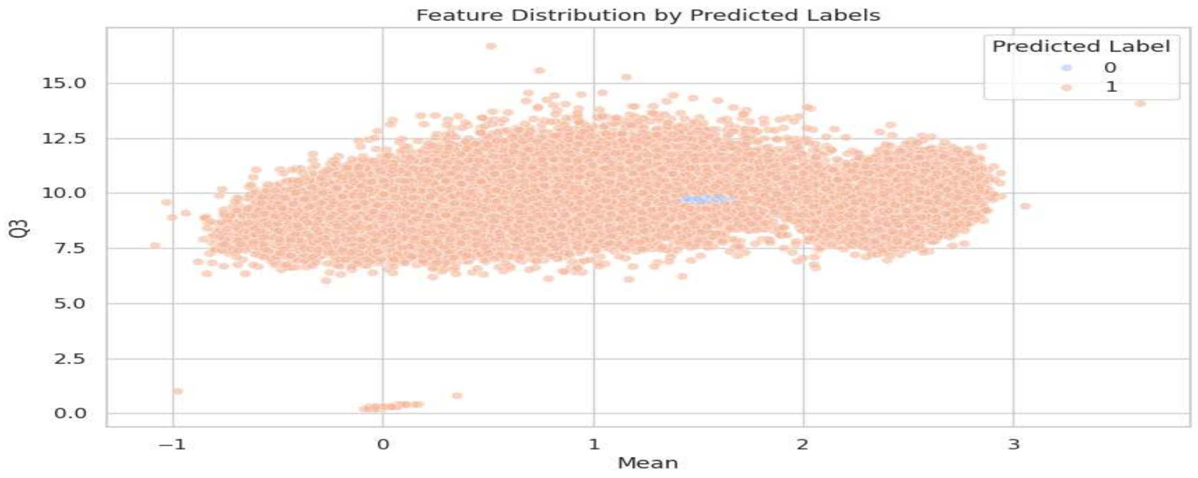


Figure 4.22: Feature distribution graph for statistical properties mean and Q3

was achieved with the combination of mean, standard deviation, minimum, and maximum. Using these optimal statistical properties, we applied the SVM classifier to all three AES implementation trace sets—unprotected, masked1, and masked2—and successfully classified them with high accuracy.

We applied the SVM classifier on the dataset using the statistical properties mean,std,min and max to classify the power traces into 3 AES implementation namely unprotected(0),masked1(1) and masked2(2). Fig 4.26 and 4.27 shows the feature distribution by class and confusion matrix. The findings are as follows:

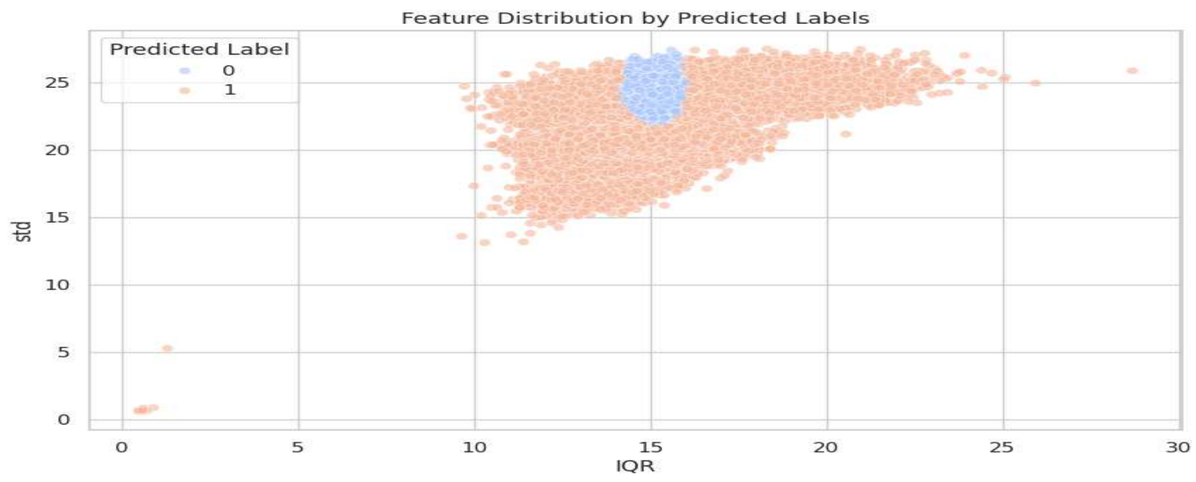


Figure 4.23: Feature distribution graph for statistical properties IQR and std

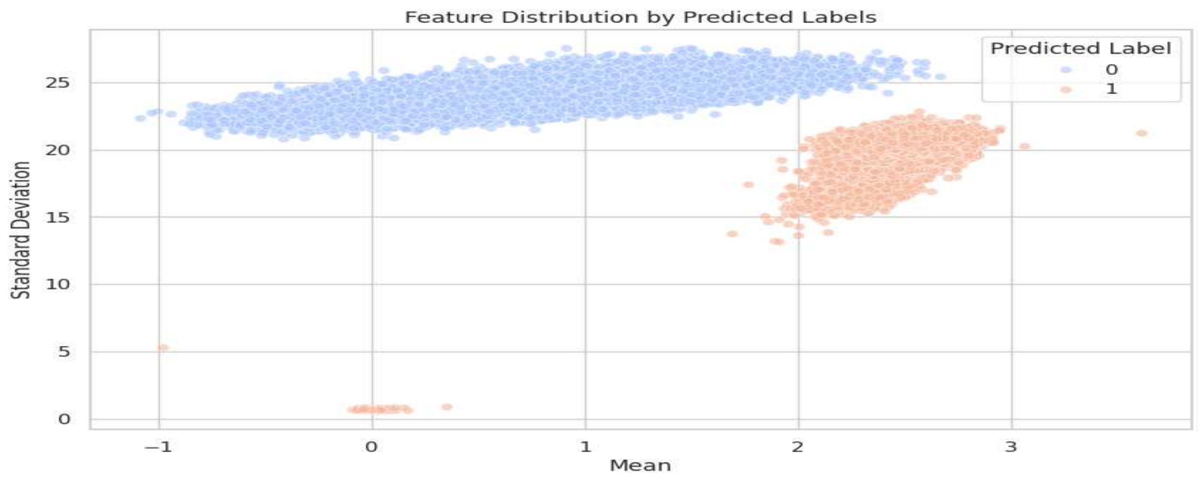


Figure 4.24: Feature distribution graph for statistical properties mean,std,min and max

Class-Based Feature Distributions

The distribution of the chosen statistical attributes (mean, standard deviation (std), minimum (min), and maximum (max)) across the three AES implementations (unprotected, masked1, and masked2) is depicted in the scatter plot matrix. Different colours are used to represent each class: Purple (0): Unprotected Teal (1): Masked1 Yellow (2): Masked2 These statistical characteristics are useful in differentiating between the various AES implementations, as the plots show clear grouping of data points for every class. The distinct distinction between clusters indicates that the chosen features have a high potential for classification.

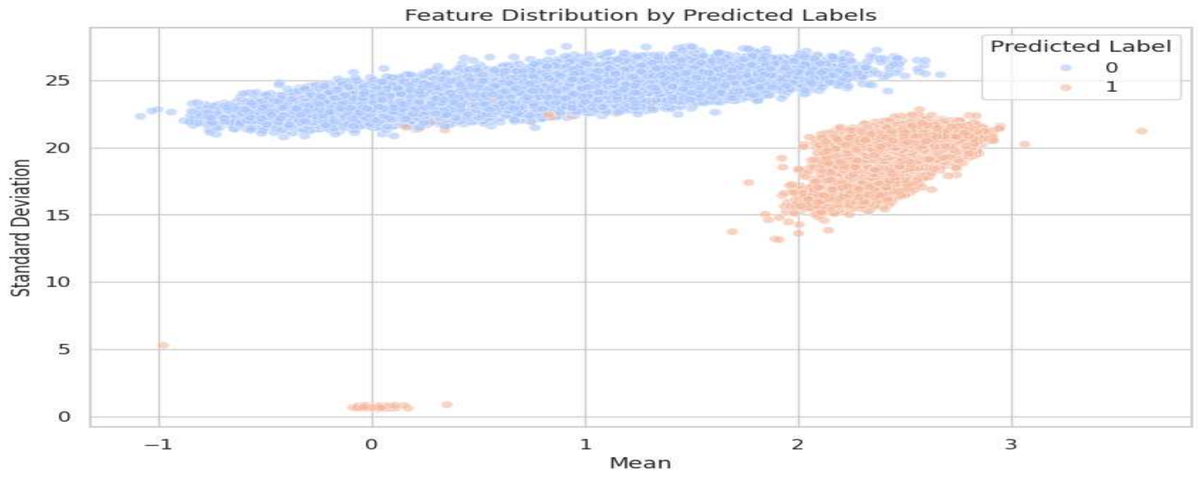


Figure 4.25: Feature distribution graph for statistical properties mean,median,min and max

Confusion Matrix

True Label 0 (Unprotected): Every one of the 20,000 real cases was correctly categorised as class 0. True Label 1 (Masked1): Of the 30,104 real cases, 30,083 were appropriately categorised, and there were only 21 cases that were incorrectly assigned to class 2. True Label 2 (Masked2): 35,083 real cases were all appropriately categorised as class 2.

Optimal Threshold Calculation and Retracing the Power Traces

ROC Curve and Optimal Thresholds

The ROC curve (Receiver Operating Characteristic curve) and AUC (Area Under the Curve) are used to evaluate the classifier's performance across all classes. The ROC curves for all classes are shown, and the AUC values are:

- Class 0 (Unprotected): AUC = 1.0
- Class 1 (Masked1): AUC = 0.9999
- Class 2 (Masked2): AUC = 0.9999

The optimal thresholds calculated for each class are:

- Class 0: 2.2238

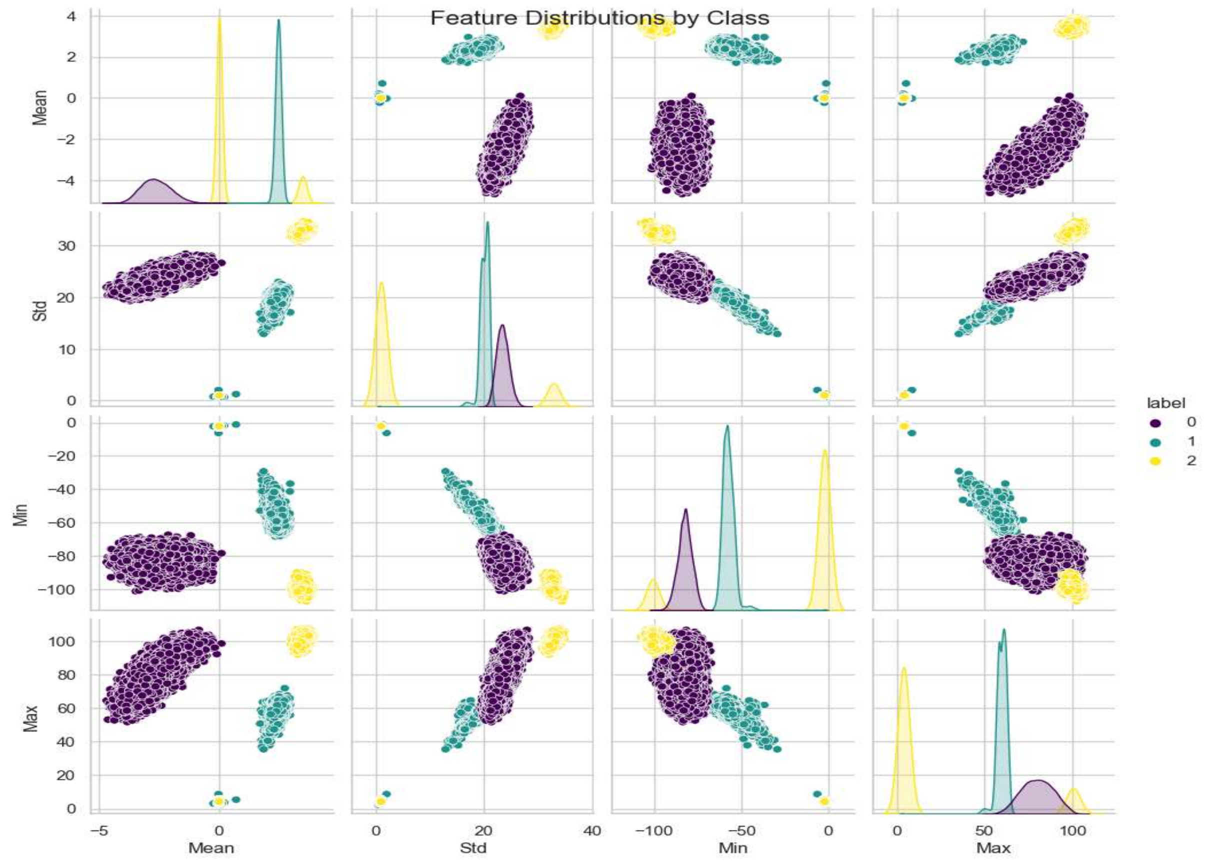


Figure 4.26: Feature distribution graph for statistical properties mean,median,min and max in classifying all implementations

- Class 1: 1.0061
- Class 2: 2.2222

The ROC curve and AUC values indicate that the classifier performs exceptionally well, with near-perfect separation between the classes. The optimal thresholds further refine the decision boundaries, ensuring precise classification. The ROC curve is shown in the Fig. 4.28

Analysis after hashing and ID method Fig 4.29 and 4.30 shows the box plot and density plots of feature distributions for train and test sets for hashing and ID method respectively. **Box plot analysis** Train Set:

- Mean: Centered around zero with minimal variation.

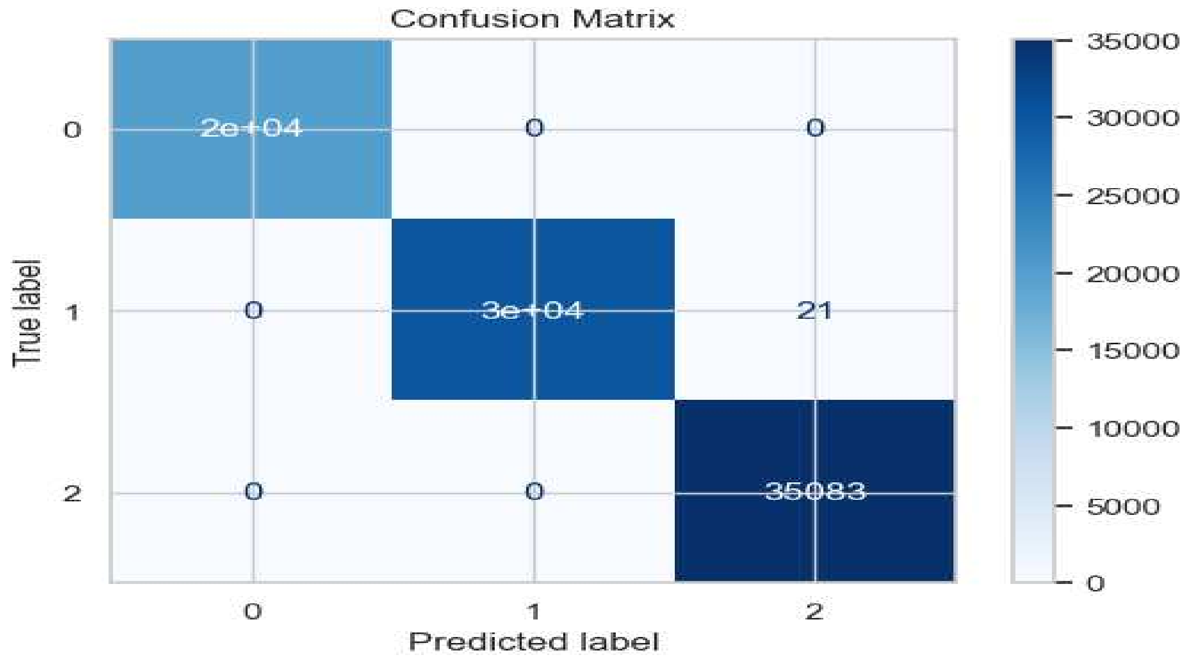


Figure 4.27: Confusion matrix for classifying AES implementations

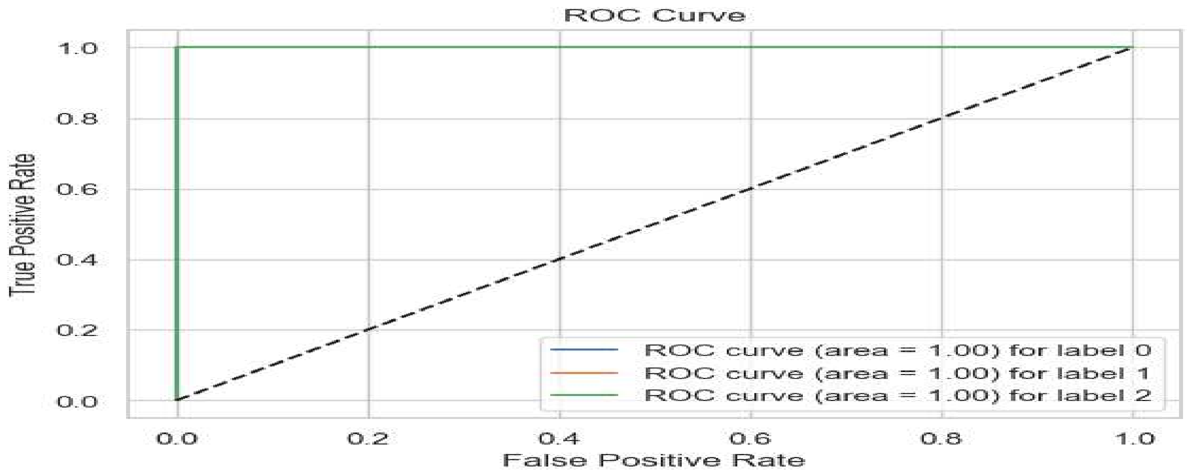


Figure 4.28: The ROC curve

- Standard Deviation: Slight dispersion with some outliers.
- Minimum: Broad range, including significant negative values.
- Maximum: Positively skewed values.

Test Set: Similar patterns to the train set, indicating consistency in data split.

- Mean: Centered around zero.
- Standard Deviation: Similar spread to the training set.
- Minimum: Broad range, similar to the train set.
- Maximum: Positively skewed.

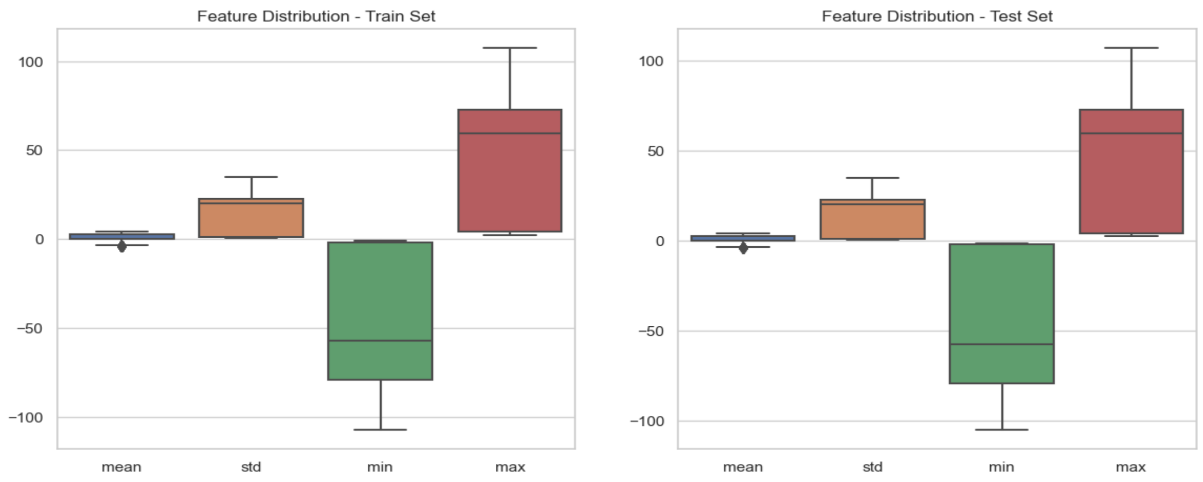


Figure 4.29: Box plot of feature distributions for train and test sets for hashing

Density peak analysis

Train Set:

- Density Peaks: Distinct peaks for each class, indicating good differentiation.
- Class Overlap: Minimal overlap, suggesting effective feature separation.

Test Set: Similar distinct peaks and minimal overlap as the train set, confirm consistent feature distributions. Overall, the features (mean, std, min, max) effectively differentiate between classes with consistent distributions in both train and test sets, supporting reliable model performance.

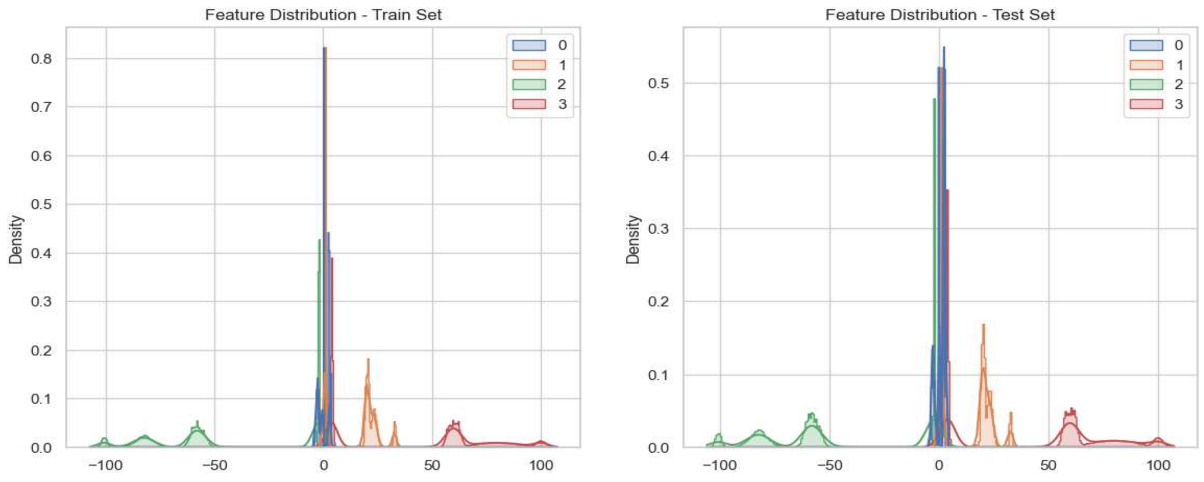


Figure 4.30: Density plots of feature distributions for train and test sets for ID method

4.4 Comparitive analysis with existing state of the art

A comparative analysis of our proposed work in module 2 with the existing techniques in the classification of power traces, feature extraction, and retracing the power traces using AES implementation is shown in Table 6.1.

- **Papers [102],[103], and [104]:** These papers leveraged the AESPT dataset and employed Estimation of distribution algorithms(EDA) power analysis models for profiling. However, they did not implement feature extraction or classification and retracing techniques.
- **Proposed Work:** Our analysis also utilized the AESPT dataset but incorporated deep learning models such as ResNet and GRUTrace. Unlike previous studies, we implemented efficient feature extraction and classification methods, along with retracing techniques like hashing and the ID method.

In conclusion, our work integrates comprehensive analytical methods and advanced deep learning models to achieve better accuracy and performance for both classification and retracing of power traces.

Table 4.1: Comparative analysis of Module 2 with existing work

Paper	**Dataset**	**Attack type**	**Model**	**Feature Extraction**	**Classification and Retracing**
[102]	AESPT	Profiling	EDA-Based PA	No	No
[103]	AESPT	Profiling	EDA-Based PA	No	No
[104]	AESPT	Profiling	EDA-Based PA	No	No
Proposed work	**AESPT**	**Profiling**	**ResTraceNet, GRUTrace**	**Yes**	**Yes**

4.5 Summary

In this module, we applied various feature extraction techniques to the statistical properties of power traces to improve the classification accuracy of AES implementations. Initially, we explored several feature extraction methods including permutation importance, recursive feature elimination, L1 regularization, mutual information, and Shapley additive explanation (SHAP). These techniques allowed us to identify the most significant features contributing to the classification performance.

To further enhance the accuracy, we employed Principal Component Analysis (PCA). Through PCA, we selected the optimal combination of features: minimum, maximum, standard deviation, and median. Using these features, we classified the power traces into three categories: unprotected, masked1, and masked2. The application of PCA was instrumental in reducing dimensionality while retaining the most informative features, which significantly improved the classification results.

The Support Vector Machine (SVM) classifier was utilized to categorize the power traces based on the extracted features. The classifier's performance was evaluated using confusion matrices and ROC curves, confirming high accuracy in distinguishing between the different AES implementations. The optimal threshold for each class was determined using ROC and AUC metrics, which ensured the classifier's reliability and precision.

Finally, we retraced the original power traces from the derived dataset using two techniques: hashing and the ID method. The hashing technique provided robust data integrity and traceability by generating unique identifiers for each data point. The ID method, on the other hand, offered a straightforward approach to tracking data points throughout the process. Both methods proved effective in ensuring accurate classification and retracing of the power traces.

Overall, our approach combining feature extraction techniques, PCA, and robust classification methods demonstrated a comprehensive strategy for accurately classifying and retracing power traces in AES implementations. The results highlight the importance of selecting optimal features and employing advanced analytical techniques to achieve high classification performance and data integrity.

Chapter 5

Module 3: Key deduction using ResTraceNet and GRUTrace

This module explores various deep learning (DL) techniques to deduce cryptographic keys from power traces in an AES implementation. We chose the ASCAD dataset, which contains power traces from an ATMEGA boolean masked AES implementation. We employed and evaluated the performance of two major DL models: ResTraceNet and GRUTrace. Additionally, we investigated the use of only 100 power traces to evaluate if sensitive information could still be deduced. Figure 5.1 provides an overview highlighting Module 3 of the research contributions.

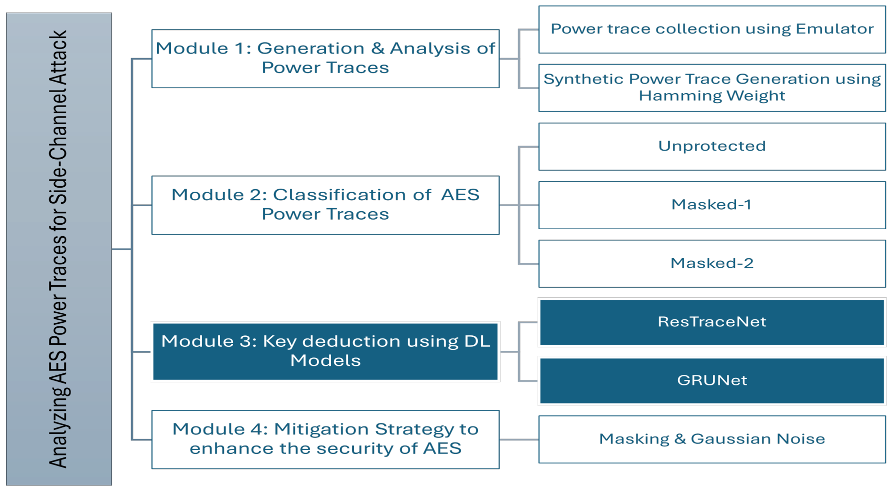

Analyzing AES Power Traces for Side-Channel Attack
Module 1: Generation & Analysis of Power Traces
Power trace collection using Emulator
Synthetic Power Trace Generation using Hamming Weight
Module 2: Classification of AES Power Traces
Unprotected
Masked-1
Masked-2
Module 3: Key deduction using DL Models
ResTraceNet
GRUNet
Module 4: Mitigation Strategy to enhance the security of AES
Masking & Gaussian Noise

Figure 5.1: Research Contributions: Highlighted Module 3

5.1 Dataset

5.1.1 Problem with AES_PTV2 dataset

When we performed the initial analysis on the AES_PTv2 dataset, we encountered several issues. The major problem was the absence of the ciphertext field, which is crucial for key deduction. Although the dataset's file structure listed a ciphertext data, no such file was present, hindering our analysis. Without the ciphertext, it is impossible to perform the necessary cryptographic computations and validate the model's effectiveness in deducing the key. Consequently, we decided to choose the ASCAD dataset as an alternative, which is ideal for key deduction.

5.1.2 ASCAD Dataset

The ASCAD (ANSSI SCA Database) dataset is a comprehensive data collection designed to act as a benchmark for the side-channel attacks community, similar to the role that the MNIST dataset plays in evaluating classification algorithms. This dataset contains EM and power traces collected from an ATMEGA microcontroller executing a boolean masked AES implementation using a fixed key. The dataset structure is divided into two major sections: profiling and attack phases, as shown in Fig. 5.2.

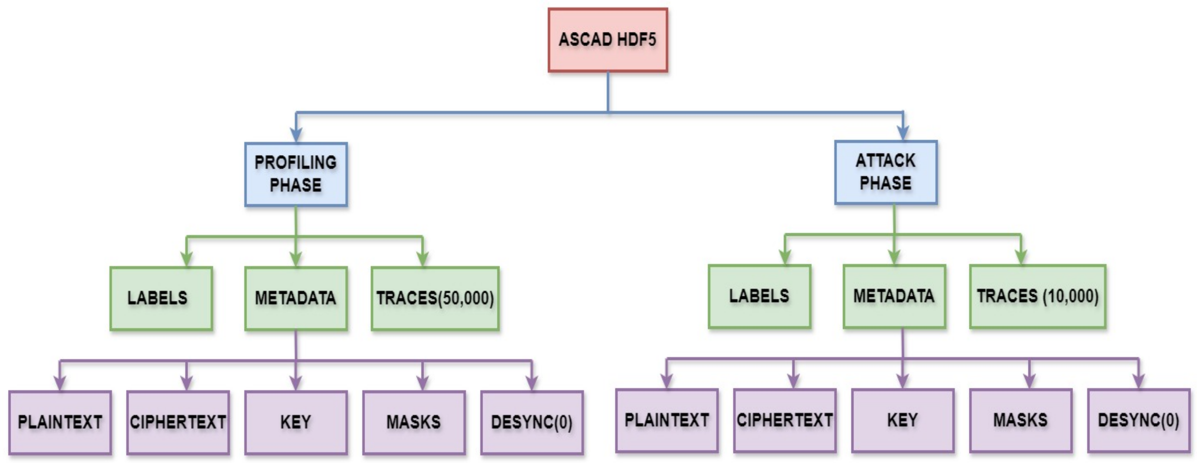


Figure 5.2: The ASCAD dataset hierarchical structure

Data Structure

The dataset is structured into two main phases:

- Profiling Phase: This phase provides a large number of traces along with the corresponding secret key, which allows researchers to build and train their models. This phase is analogous to a training dataset in machine learning.

- Attack Phase: In this phase, traces are provided without the secret key, challenging the models to predict the key. This phase serves as the test set for evaluating the effectiveness of the trained models.

Profiling Phase

- Labels: Target labels output used for training the model, which are S-Box outputs.
- Metadata: Contains integral data such as plaintext, ciphertext, key, masks, and desynchronization (intentional misalignment of power traces to simulate real-world traces).
- Power Traces: Actual raw power traces collected while running the AES implementation.

Attack Phase

- Labels: Used to evaluate the performance of the profiling phase model during the attack phase.
- Metadata: Contains integral data such as plaintext, ciphertext, key, masks, and desynchronization, similar to the profiling phase.
- Power Traces: Actual raw power traces collected to attack the AES implementation.

Data Formats

The dataset is often provided in CSV or HDF5 formats, making it accessible for data processing and analysis using various programming languages and tools.

Applications

The ASCAD dataset is widely used in research to:

- Develop and evaluate side-channel attack techniques, such as Differential Power Analysis (DPA), Correlation Power Analysis (CPA), and template attacks.

- Test the effectiveness of various machine learning and deep learning models, including CNNs,RNNs and others, in predicting cryptographic keys from side-channel traces.

- Study the effectiveness of countermeasures, like masking and noise addition, in protecting cryptographic implementations.

Challenges and Considerations

- Desynchronization and Noise: Real-world traces often include desynchronization and noise, which complicates the analysis. The ASCAD dataset sometimes includes such factors to better simulate realistic scenarios.

- High Dimensionality: The traces are often high-dimensional, presenting challenges in terms of data processing and model training, especially when using deep learning techniques.

- Ethical Considerations: While the dataset is invaluable for advancing the field, it also underscores the need for ethical considerations in cybersecurity research, particularly in the responsible disclosure and mitigation of vulnerabilities.

This dataset is built to facilitate the training and evaluation of machine learning and deep learning models in side-channel analysis, providing essential information for both model building and the attack phase.

5.2 Methodology

In this section, we leverage powerful deep learning models, specifically ResTraceNet and GRUTrace, to predict the S-Box output and subsequently deduce the cryptographic key of a masked AES implementation using the ASCAD dataset. We use the profiling phase data to train the model and then apply the trained model to the attack phase data for key deduction. The overall methodology workflow is depicted in Figure 5.3. This diagram illustrates the complete process from data preparation and feature engineering to model training and key deduction as explained below:

- **Data Preparation and Feature Engineering** This initial phase involves converting the hierarchical HDF5 file into csv files, loading the power traces, associated metadata, and labels from the ASCAD dataset. Pre-processing steps include decoding the cipher text from hexadecimal to a numeric format suitable for analysis, scaling the power traces to standardize the data, and applying Principal Component Analysis (PCA) to reduce dimensionality and focus on the most informative features. These processed components are then concatenated with additional metadata to shape and categorize the data effectively for subsequent modeling.

- **Model Training with ResTraceNet and GRUTrace** In this section, the architecture of both ResTraceNet and GRUTrace models is set up to handle the characteristics of the time-series data provided by the power traces. The models are trained on the prepared dataset, utilizing techniques such as "early stopping" to prevent overfitting and "ReduceLROnPlateau" to adjust the learning rate dynamically based on performance metrics. This systematic training helps in optimizing the models' ability to predict the correct S-box outputs from the AES implementation.

- **Model Evaluation and Prediction** Once models are trained, they utilize a "softmax" output layer to transform the logits into a probability distribution across 256 possible classes, each corresponding to a potential key byte. The model's performance is rigorously evaluated through accuracy assessments and the generation of confusion matrices. These tools help in quantifying the effectiveness of the models in predicting the correct cryptographic keys.

- **Key Deduction from S-Box Output** The final phase involves applying the trained models to new, unseen test data to predict the S-box outputs. These predictions undergo inverse S-box mapping to deduce the possible key values. The results from multiple models and predictions are aggregated to ascertain the most likely cryptographic key, providing a robust conclusion to the key deduction process. We also use 100 power traces and apply the most optimal models to demonstrate their power in predicting S-box output and key.

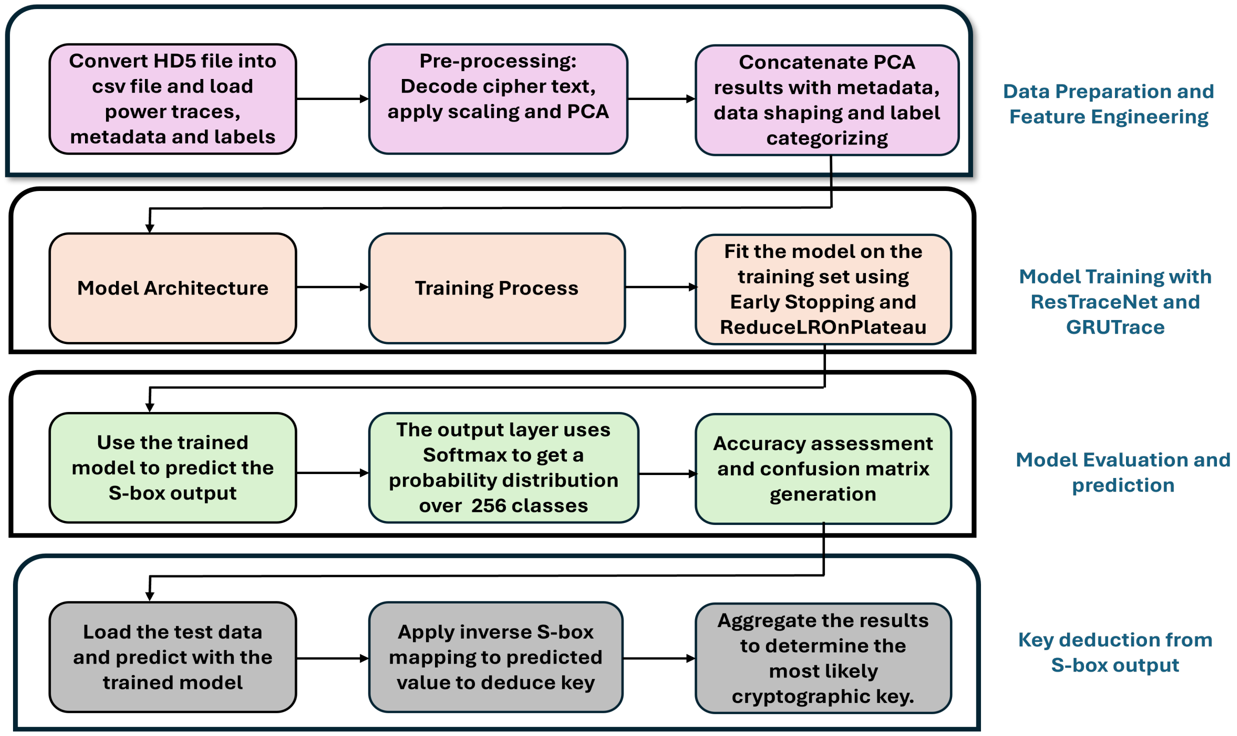

Convert HD5 file into csv file and load power traces, metadata and labels
Pre-processing: Decode cipher text, apply scaling and PCA
Concatenate PCA results with metadata, data shaping and label categorizing
Data Preparation and Feature Engineering
Model Architecture
Training Process
Fit the model on the training set using Early Stopping and ReduceLROnPlateau
Model Training with ResTraceNet and GRUTrace
Use the trained model to predict the S-box output
The output layer uses Softmax to get a probability distribution over 256 classes
Accuracy assessment and confusion matrix generation
Model Evaluation and prediction
Load the test data and predict with the trained model
Apply inverse S-box mapping to predicted value to deduce key
Aggregate the results to determine the most likely cryptographic key.
Key deduction from S-box output

Figure 5.3: Overall Methodology Workflow

5.2.1 ResTraceNet: ResNet-Inspired Architecture for Power Trace Analysis

In this research, we propose ResTraceNet, an adaptation of the ResNet architecture tailored for power trace data analysis. ResTraceNet employs 1-dimensional convolutions organized into blocks inspired by the ResNet framework. Each block is designed to capture temporal dependencies within the data, similar to how ResNet blocks capture spatial features in images. We explore the effectiveness of ResTraceNet on ASCAD dataset, demonstrating its ability to deduce the AES key.

1. Resnet Adaptation for Power Trace Data: ResTraceNet is a specialized version of the traditional ResNet model, redesigned to handle the requirements of analyzing one-dimensional power trace data, which is commonly used in time-series analysis. Unlike the original ResNet that is tailored for images and uses

2D convolutional layers, ResTraceNet uses 1D convolutional layers. These layers are effective at detecting patterns over time, making them ideal for analyzing sequences of data recorded during cryptographic operations.

The core of ResTraceNet consists of several blocks, each containing 1D convolutional layers followed by normalization and activation functions. These blocks are designed to extract features progressively from the power traces, enhancing the model's ability to understand complex data patterns essential for predicting cryptographic keys.

Each block in the architecture builds on the previous one, starting with simple feature detection and advancing to more complex analyses. This layering helps in refining the model's predictions. The design includes shortcut connections that help maintain the flow of information across the network, preventing common issues like gradient vanishing and enabling deeper analysis.

Overall, the structure of ResTraceNet allows it to effectively handle the complexities of encrypted power trace data, optimizing it for tasks such as predicting S-box outputs and deducing cryptographic keys.

2. 1-Dimensional Convolutions: The ResNet architecture, initially designed for 2D image processing, has been tailored to analyze one-dimensional time-series data such as power traces. This adaptation involves replacing traditional 2D convolutional layers with 1D layers that are better suited for handling sequential data. These 1D layers effectively capture temporal patterns and features crucial for cryptographic analysis by sliding filters over the data sequence.

   The structure of the ResTraceNet, now focused on 1D convolutions, incorporates blocks of these layers followed by normalization and activation functions. These blocks are designed to progressively deepen the analysis without information loss, supported by shortcut connections that prevent gradient vanishing. This design is repeated several times within the architecture to handle increasing complexity and extract more abstract features.

The implementation of this architecture involves multiple blocks of varying complexity. Training employs the Adam optimizer with dynamic learning rate adjustments and early stopping mechanisms to ensure optimal generalization on unseen data, crucial for tasks like S-Box output prediction and key deduction. An overview of the ResTraceNet architecture with each component's function and purpose is shown in Table 5.1.

Table 5.1: Overview of ResTraceNet Architecture Components

Component	**Function**	**Purpose**
1D Convolutional Layers	Utilizes filters to capture temporal patterns in one-dimensional power trace data.	Essential for detecting time-dependent features and patterns within the power traces.
Batch Normalization	Standardizes outputs of convolutional layers to improve network stability and speed.	Enhances the consistency and efficiency of the model by normalizing layer inputs.
Activation Function	Uses LeakyReLU to introduce non-linearity.	Helps the model to learn complex and non-linear patterns from the data.
Shortcut Connections	Allows gradients to flow directly through the network, preventing the vanishing gradient problem.	Supports deeper network architectures by enabling effective training of numerous layers.
Block Design	Each block contains a series of convolutional, normalization, and activation layers.	Facilitates progressive refinement of features, enhancing the network's learning capability.
Significance of Blocks	Designed to incrementally extract and refine features, from basic to complex.	Critical for building a deep understanding of the data, leading to more accurate predictions.
Repetition Pattern	Blocks are repeated based on the complexity required by the task.	Allows the network to progressively enhance its feature extraction capabilities and accuracy.

5.2.2 ResTraceNet model

We employed and evaluated three variants of the ResTraceNet model to determine the most effective architecture for predicting the S-box output and then deducing the key. Initially, the models were trained using the profiling phase data, and based on their performance, we identified the most optimal model. Once the best-performing variant was determined, we used it on the attack dataset to glean sensitive information.

ResTraceNet Variant 1

We started by loading the dataset containing power traces, metadata, and labels from the CSV files. The power traces were scaled using the StandardScaler, and the dimensionality was reduced to 50 PCA components. This pre-processing phase ensures that the features fed into the model are well-normalized and less complex, which facilitates more efficient training.

Architecture

The architecture consists of an input layer that receives the preprocessed power trace data. This input data is then passed through a series of convolutional and residual blocks designed to capture intricate relationships and features in the power trace.

The first layer is a 1D convolutional layer with 128 filters and a kernel size of 7, followed by multiple residual blocks. Each residual block contains two convolutional layers with batch normalization and Leaky ReLU activation functions. Identity shortcut connections connect each block's input directly to the output, facilitating gradient flow during backpropagation and enabling deeper networks without vanishing gradient issues. Some residual blocks include downsampling, achieved by using a stride of 2 in the convolutional layers, to reduce dimensionality and capture hierarchical features.

A global average pooling layer is added to condense the feature maps into a single vector, and a dropout layer with a rate of 0.5 is included to prevent overfitting. The final dense layer uses a softmax activation function to output the probabilities for each of the 256 possible S-box values, enabling the model to deduce the key.

ResTraceNet Variant 2

In ResTraceNet Variant 2, we aimed to improve the efficiency of Variant 1 by making changes in the model architecture and hyperparameters. The architecture began with an input layer that received the processed power traces and metadata, followed by convolutional and residual blocks designed to capture hidden patterns. The initial convolutional layer used 128 filters with a kernel size of 7 and a stride of 2.

This variant used numerous residual blocks, each consisting of two convolutional layers, batch normalization, and ReLU activation functions. Identity shortcut connections were employed to counter the vanishing gradient problem and enable gradient flow in each residual block. Some residual blocks included downsampling, which utilized a stride of 2 in the convolutional layers, reducing dimensionality and capturing hierarchical features.

The architecture also included a global average pooling layer, which condensed the feature maps into a single vector, making it appropriate for the dense output layer. This was followed by a dropout layer with a rate of 0.2 to prevent overfitting. The final dense layer employed a softmax activation function to output probabilities for each of the 256 possible S-box values, enabling the model to accurately predict the S-box output and deduce the cryptographic key from the power traces. Additionally, a learning rate scheduler was implemented to adjust the learning rate dynamically, improving the training phase.

ResTraceNet Variant 3

This ResTraceNet architecture consists of a total of 12 convolutional layers distributed across 6 residual blocks, along with other essential layers such as BatchNormalization, LeakyReLU, GlobalAveragePooling, and Dropout. This architecture is designed to handle one-dimensional data, specifically power traces, which are sequences of power measurements taken over time. The use of 1D convolutional layers allows the model to effectively capture temporal patterns and features from the power trace data, facilitating accurate prediction of S-box outputs and key deduction.

The architecture starts with an input layer that is configured to receive one-dimensional data of length input_dim. This input data represents power traces collected over time during AES cryptographic operations.

The first layer is a 1D convolutional layer with 128 filters and a kernel size of 7. It uses a stride of 2, which helps in downsampling the input data and capturing broader features. This layer is crucial for extracting initial patterns from the raw power trace data.

Following the initial convolutional layer, the architecture includes a series of six residual blocks. Each residual block is designed to improve feature learning through shortcut connections. The blocks are structured as follows:

- **Residual Block 1:** Comprising two 1D convolutional layers, each with 128 filters and a kernel size of 3. Batch normalization and LeakyReLU activation are applied after each convolution. This block maintains the same spatial dimensions as the input.

- **Residual Block 2:** Similar to the first block but includes downsampling. The convolutional layers have a stride of 2, effectively reducing the spatial dimensions and allowing the model to capture hierarchical features. The identity connection is also adjusted to match the new dimensions.

- **Residual Block 3:** Comprising two 1D convolutional layers with 256 filters and a kernel size of 3. Batch normalization and LeakyReLU activation are again applied after each convolution. This block captures more detailed features.

- **Residual Block 4:** Similar to the third block but includes downsampling with a stride of 2.

- **Residual Block 5:** Comprising two 1D convolutional layers with 512 filters and a kernel size of 3. Batch normalization and LeakyReLU activation continue to be used.

- **Residual Block 6:** Similar to the fifth block but includes downsampling with a stride of 2.

After the series of residual blocks, a global average pooling layer is used. This layer condenses each feature map into a single value by averaging all its elements. This reduction simplifies the data while retaining the most important features learned by the convolutional layers.

Following global average pooling, a dropout layer with a rate of 0.3 is included. Dropout is a regularization technique that helps prevent overfitting by randomly setting a fraction of the input units to zero during training.

The final layer is a dense layer with a softmax activation function. This layer outputs a probability distribution over the 256 possible classes (S-box values), enabling the model to classify the input data accurately.

For training, the model uses the Adam optimizer, which adjusts the learning rate dynamically to improve convergence. A learning rate scheduler is employed to adjust the learning rate value dynamically during training, starting at 0.001 for the initial 10 epochs and then reducing it to 0.0001 for the remaining epochs. This method helps stabilize the training process and achieve optimal convergence. Early stopping is employed to monitor the validation loss and halt training if no improvement is observed for 20 epochs, restoring the best model weights. Additionally, a learning rate scheduler is used to decrease the learning rate over time, which helps in achieving optimal convergence.

This architecture, with its deep residual learning framework and 1D convolutional layers, is designed to handle the complexities of power trace data effectively, making it suitable for tasks such as S-box output prediction and cryptographic key deduction in AES implementations.

Fig. 5.4 shows the architecture of ResTraceNet variant 3.

The Table 5.2, summarizes the hyperparameters and performance metrics of the three ResTraceNet variants. Each model has a unique configuration concerning dropout rates, L2 regularization, learning rate schedules, batch sizes, activation functions, and early stopping criteria.

Attribute	**Model 1**	**Model 2**	**Model 3**
Dropout Rate	0.5	0.2	0.2
L2 Regularization	0.0005	0.0005	0.0005
Learning Rate Schedule	Cosine Annealing with Warm Restarts	Step Decay: 0.001, 0.0005, 0.0001, 0.00005	Step Decay: 0.001, 0.0005, 0.0001
Batch Size	64	128	64
Activation Function	LeakyReLU (0.01)	ReLU	LeakyReLU (0.01)
Epochs	200	150	200
Early Stopping Patience	20	30	30
Additional Layers	Standard ResNet	Standard ResNet	Standard ResNet
Optimizer	Adam (initial LR=0.001)	Adam (initial LR=0.001)	Adam (initial LR=0.001)
PCA Components	50	50	50

Table 5.2: Comparison of different ResTraceNet variants in hyperparameters and performance metrics

5.2.3 GRUTrace: Adaptation of GRU for Analyzing Power Trace Data

In adapting the traditional GRU architecture to analyze power trace data, the developed model, termed GruTrace, retains the foundational structure of a Recurrent Neural Network (RNN). This adaptation leverages the inherent capabilities of RNNs, particularly suited for sequential data analysis, by implementing Gated Recurrent Units (GRUs) in a bidirectional and layered configuration. Here's how these modifications were implemented:

1. **Bidirectional GRU Layers:** The GRU cells were configured in a bidirectional setup, enabling the network to process data both forwards and backwards. This dual-direction processing is essential for capturing dependencies that provide a comprehensive view of the data's context, which is crucial for accurate cryptographic analysis.

2. **Layer Normalization:** To enhance the stability and efficiency of the training process, layer normalization was integrated within each GRU layer. This normalization ensures that the outputs across different layers are on a similar scale, which is vital given the variability and potential noise present in power trace data.

3. **Integration of Additional Features:** Recognizing the value of contextual information, the input layer was modified to include additional data dimensions such as metadata. This adaptation allows the model to utilize a richer set of inputs, significantly enhancing its ability to make informed predictions.

4. **Dropout for Regularization:** To prevent the model from overfitting to the noisy training data, dropout regularization was strategically implemented within the network. This technique involves randomly disabling a fraction of the neurons during training, which encourages the development of more robust features that generalize better to unseen data.

This adapted GRUTrace architecture demonstrates the flexibility of deep learning techniques to meet the demands of highly specialized data analysis tasks, maximizing both performance and applicability in complex domains like cryptographic analysis.

5.2.4 GRUTrace model

This architecture is designed to work efficiently with sequential data, making it ideal for side-channel attacks where power traces are treated as time-series data. The Fig. 5.5 shows the architectural diagram of the GRUTrace model.

- **Data Loading and Preprocessing**

  The dataset, which contains power traces, metadata, and labels, is loaded from CSV files. The ciphertext in the metadata is decoded from its hexadecimal representation into numerical values. To prepare the data, the power traces are standardized using the StandardScaler to normalize the values. Principal Component Analysis (PCA) is then applied to reduce the dimensionality to 50 components, capturing the highest variance in the power traces. The power traces are combined with the ciphertext and the third byte of the plaintext to create the complete feature set, with any missing values filled with zero.

- **Model Architecture**

  The model begins with an input layer designed to receive data shaped according to the number of features, including the PCA components, plaintext, and ciphertext. The core of this model consists of three bidirectional GRU layers. The first bidirectional GRU layer has 128 units and processes input data in both forward and reverse directions, capturing dependencies across the entire sequence of data. After this layer, we apply layer normalization to stabilize the output, followed by a dropout layer with a rate of 0.3 to prevent overfitting.

  The second layer, also bidirectional, contains 256 units. Like the first layer, it returns sequences and is followed by layer normalization and a dropout layer. The third bidirectional GRU layer, with 128 units, summarizes the information collected from the previous layers into a single output without returning sequences.

  The final output is passed through a dense layer with a softmax activation function, producing a probability distribution over the possible 256 values. The model is compiled using the Adam optimizer and categorical cross-entropy loss, which is ideal for multiclassification tasks. Early stopping is implemented to halt training if the validation loss does not improve for 20 epochs, and learning rate reduction on plateau adjusts the learning rate if the validation loss stagnates. The model is trained for 150 epochs with a batch size of 64.

This GRUTrace model architecture, with its bidirectional layers and regularization techniques like layer normalization and dropout, is designed to leverage the sequential nature of power traces effectively.

5.2.5 Testing on attack phase data and predicting Sbox output

During the testing phase, we utilize the trained ResTraceNet and GRUTrace models to predict the S-Box output from the AES implementation and subsequently deduce the cryptographic key. Initially, we load the trained models and the test data, which includes power traces, labels, and metadata. The metadata is processed to extract the ciphertext values. Consistency is ensured by applying the same scaling and PCA transformations used during training.

Once the test data is preprocessed, it is reshaped to align with the input requirements of the chosen model. The model then predicts the S-Box output classes for each trace in the dataset. The predicted classes are compared with the actual classes to evaluate the model's performance on the test set.

5.2.6 Key deduction

The key deduction process begins after predicting the S-Box output. This involves using the predicted S-Box values and the plaintext to deduce the most likely key byte. An inverse S-Box is used to identify possible key candidates. The frequency of each candidate is recorded, and the candidate with the highest frequency is deemed the most probable key byte. This method helps in accurately identifying the correct key byte with high confidence. To visualize the results, histograms, line charts, and pie charts are generated to display the distribution of key candidates and the confidence level in the most likely key byte.

5.2.7 Key Deduction Using ResTraceNet and GRUTrace Models with 100 power traces

In this section, we leverage only 100 power traces and metadata to extract the AES key byte. The ResTraceNet variant 3 and GRUTrace models used in the previous section are employed to predict the S-Box output from the power traces. The architecture of the models remains consistent with the previously explained designs. The most optimal versions of ResTraceNet are used for this prediction. By utilizing the capabilities of these deep learning models, we aim to accurately deduce the cryptographic key from the 100 power traces.

5.3 Results and discussion

In this section, we present the output obtained from training the three variants of ResTraceNet and GRUTrace models by plotting the PCA variance explained graph, training vs. validation accuracy, and training vs. validation loss graph. Furthermore, we extend this experiment to 100 power traces using the most optimal variant from the ResTraceNet and GRUTrace models.

5.3.1 ResTracenet variant1

For ResTraceNet variant 1, the PCA variance explained graph (5.6) shows that the initial few principal components contributed the most variance in the dataset, demonstrating the effectiveness of PCA in capturing the most integral features. The training vs. validation accuracy and loss graphs show a clear improvement in training accuracy but also exhibit significant variations in validation accuracy, indicating overfitting. The validation loss graph displays some variance, indicating instability during training. Fig.5.7 shows the training vs validation accuracy and training vs validation loss for ResTraceNet variant 1.

5.3.2 ResTracenet variant2

For ResTraceNet variant 2, the PCA variance explained graph (5.8 similarly indicates that the initial principal components capture a major percentage of the variance. The training and validation accuracy graph shows an increase but with high fluctuations in validation accuracy, suggesting instability. The training and validation loss graphs show a steady decline in losses, but with recurrent spikes in validation loss, reflecting overfitting and instability. Fig.5.9 shows the training vs validation accuracy and training vs validation loss for ResTraceNet variant 2.

5.3.3 ResTracenet variant3

ResTraceNet variant 3 shows the most consistent performance across all three models with high training accuracy and fewer fluctuations. The PCA variance explained graph for Variant 3 as shown in Fig. 5.10 is similar to the previous variants, indicating that the first few principal components capture the majority of the variance in the data. This efficiency in extracting crucial features is vital for the model to learn patterns clearly.

The training vs. validation accuracy graph for Variant 3 shows a gradual increase in accuracy over the epochs. The validation accuracy remains consistently high, suggesting that the model is not overfitting and can generalize well with new, unseen data. The training vs. validation loss graph demonstrates the model's stability, with a steady decline in both training and validation loss values, indicating effective learning and minimal error.

To conclude, Variant 3 is the most optimal choice for key deduction from the power traces. Fig.5.11 shows the training vs validation accuracy and training vs validation loss for ResTraceNet variant 3.

5.3.4 GRUTrace model

Using the GRUTrace model, the PCA variance explained as indicated by Fig. 5.12 is similar to all the previous models, showing that the first few components captured a large proportion of the variance. The training vs. validation accuracy graph for the GRU model indicates a steady increase, demonstrating that the model is learning effectively. The training vs. validation loss graph exhibits a smooth decline, indicating that the model is becoming better at reducing loss over epochs. This stability in both graphs further proves the GRUTrace model's effectiveness in learning from the AES power traces and accurately predicting the S-box output. Fig.5.13 shows the training vs validation accuracy and training vs validation loss for GRUTrace model.

5.3.5 Using 100 power traces with ResTraceNet and GRUTrace model

We experimented with using only 100 power traces, utilizing the ResTraceNet variant 3, which yielded the most optimal model as indicated by the graphs and results. The training vs. validation accuracy and loss graphs show a clear increase in accuracy and a steady decline in loss, even with a reduced number of traces as shown in Fig. 5.14. This demonstrates the model's capability to make accurate predictions even with limited data. We also evaluated the GRUTrace model's performance in predicting the S-box output using 100 power traces. The accuracy graphs indicate a steady increase in both training and validation accuracy, with some fluctuations in validation accuracy, suggesting slight overfitting. The loss graph demonstrates a gradual decline in both training and validation loss, showing the model's ability to minimize prediction errors over time. Although there are small variations, the overall performance suggests that the GRUTrace model successfully learns the patterns necessary for predicting the S-box output and deducing the cryptographic key. Both the accuracy and loss graph for GRUTrace are shown in Fig. 5.15

5.3.6 S-box prediction and key deduction

Using ResTraceNet variant 3, we achieved a training accuracy of 91.6% and a validation accuracy of 99.4%. The highest testing accuracy recorded for this model was 96.68%. For the GRUTrace model, the training accuracy reached 91%, with a validation accuracy of 96.8% and a slightly lower testing accuracy of 96.28%.

When using 100 power traces, the results were as follows:

The ResTraceNet model achieved a training accuracy of 82.7%, a validation accuracy of 92.13%, and a testing accuracy of 96.8%. The GRUTrace model achieved a training accuracy of 87.6%, a validation accuracy of 98.7%, and a testing accuracy of 98.53%. By leveraging ReTracesNet variant 3 and the GRUTrace model with only 100 power traces, we successfully predicted the S-box outputs and deduced the AES key byte. The original key byte array was [77, 251, 224, 242, 114, 33, 254, 16, 167, 141, 74, 220, 142, 73, 4, 105], and the deduced key byte was 224. This demonstrates the effectiveness of our proposed models in extracting sensitive information from the power traces, validating their potential in countering side-channel attacks.

Visualization

For the ResTraceNet model, we performed the following visualizations. Fig. 5.16 shows a pie chart representing the confidence in the most likely key byte, where the model predicted the key with 95.6% confidence. Fig. 5.17 indicates a histogram of key candidate frequencies, highlighting the dominance of the most likely key candidate. Fig. 5.18 shows a confusion matrix, illustrating the model's high accuracy in predicting the correct S-box values. We also performed a temporal accuracy analysis for ResTraceNet Variant 3 as shown in Fig. 5.21 This analysis demonstrates the model's consistent performance over different time steps, further validating its robustness.

For the GRUTrace model, Fig. 5.20 and 5.21 shows the line chart of key candidate frequencies and confusion matrix confirming the GRUTrace model's effectiveness in pinpointing the correct key byte.

Using 100 traces for the GRUTrace model and ResTraceNet model, we visualized the histogram of key candidate frequency and confusion matrix as indicated by Fig. 5.22

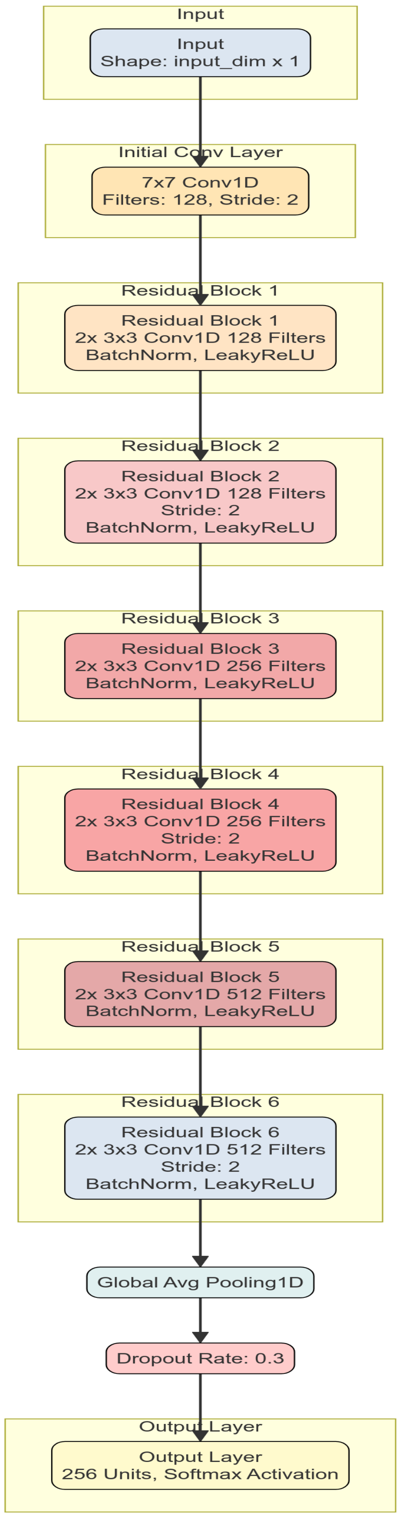


Figure 5.4: The architecture of ResTraceNet variant3

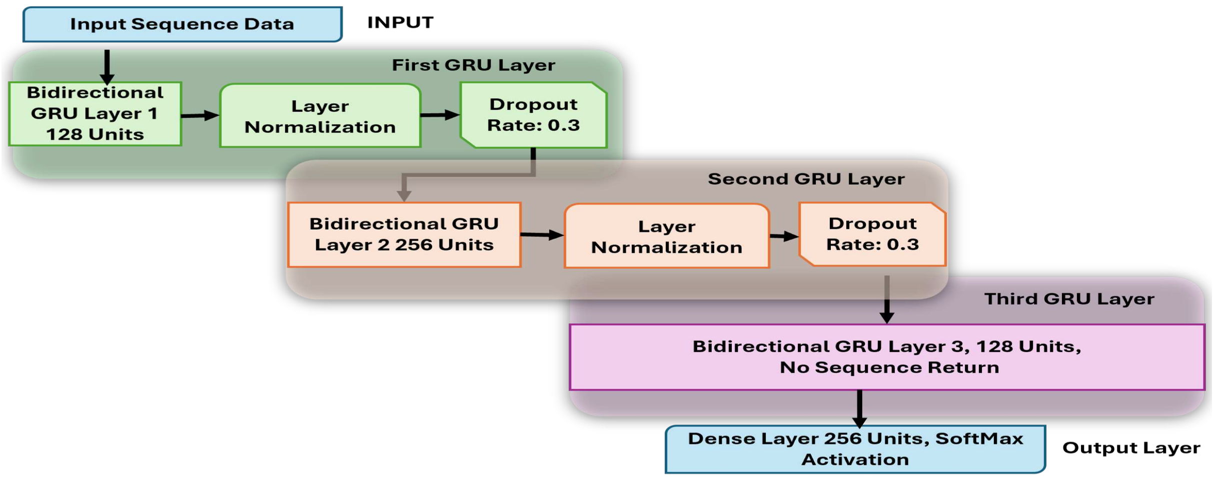

Input Sequence Data
INPUT
First GRU Layer
Bidirectional GRU Layer 1 128 Units
Layer Normalization
Dropout Rate: 0.3
Second GRU Layer
Bidirectional GRU Layer 2 256 Units
Layer Normalization
Dropout Rate: 0.3
Third GRU Layer
Bidirectional GRU Layer 3, 128 Units, No Sequence Return
Dense Layer 256 Units, SoftMax Activation
Output Layer

Figure 5.5: The architecture of GRUTrace model

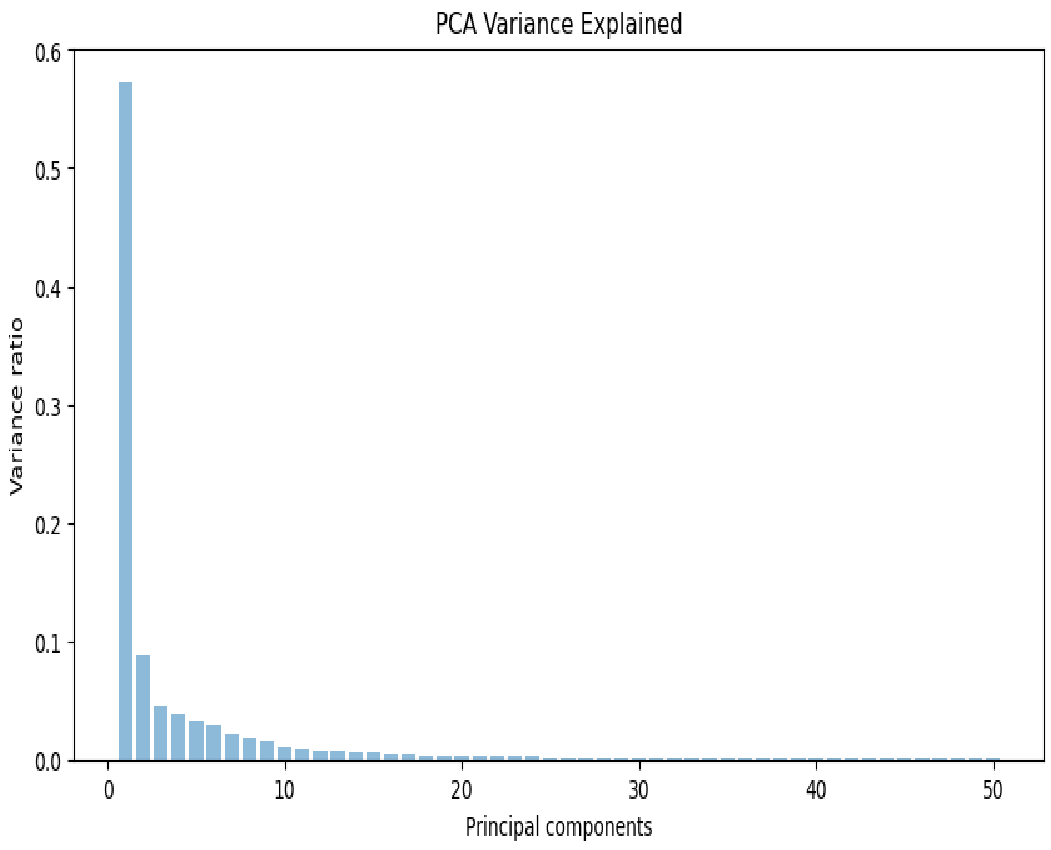
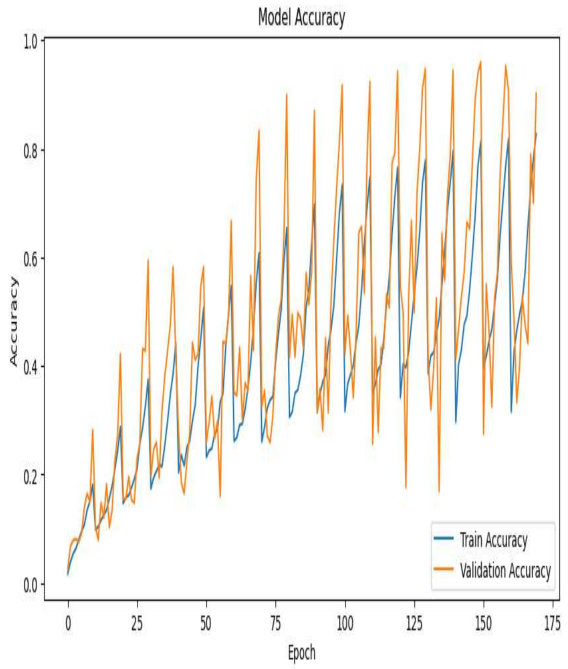
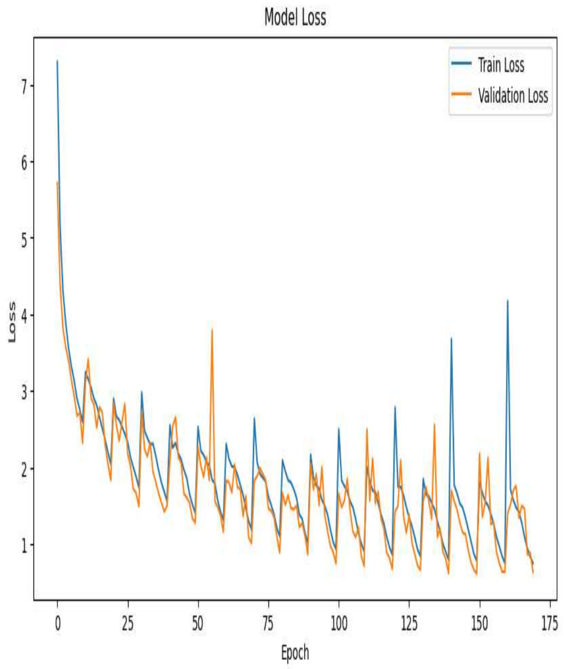


Figure 5.6: The PCA variance explained graph for ResTraceNet Variant 1



(a) Training vs validation accuracy graph for ResTraceNet variant 1



(b) Training vs validation loss graph for variant 1

Figure 5.7: training vs validation accuracy and training vs validation loss for ResTraceNet variant 1

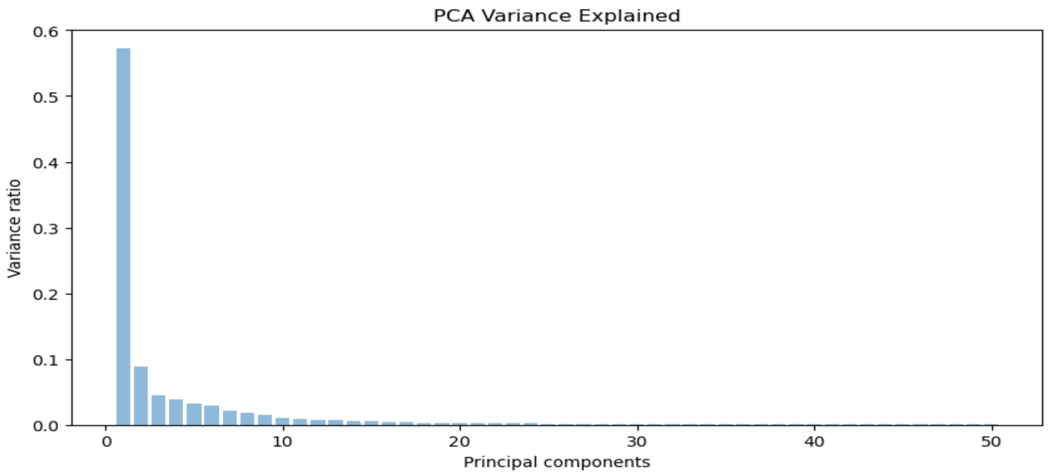
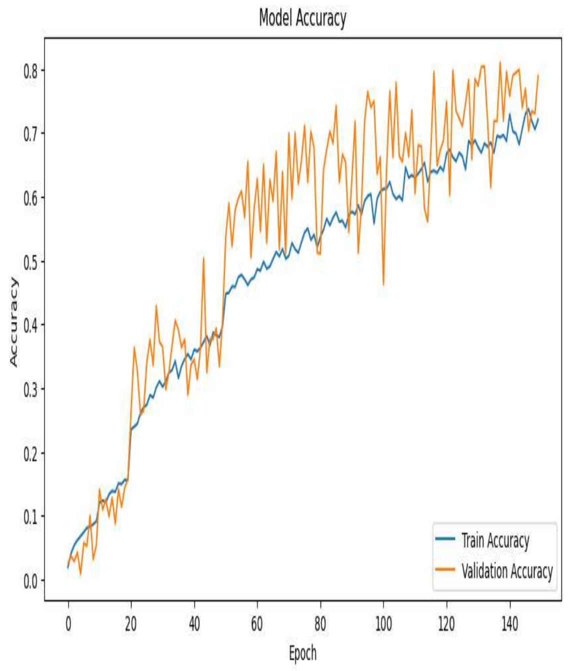
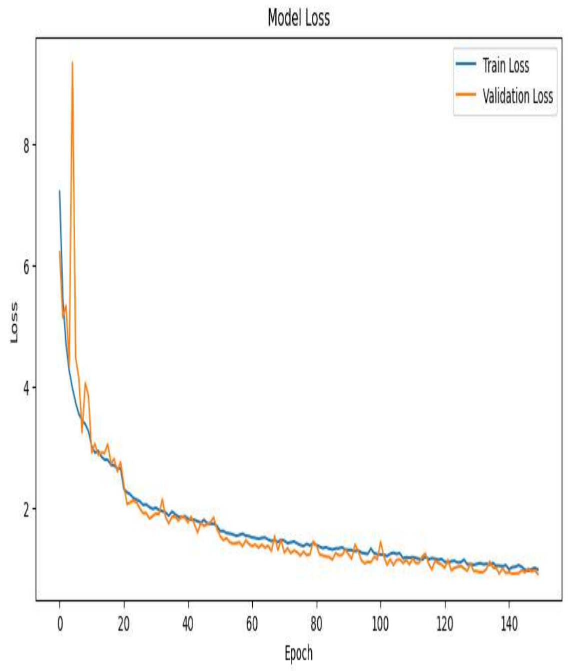


Figure 5.8: The PCA variance explained graph for ResTracenet Variant 2



(a) Training vs validation accuracy graph for variant 2



(b) Training vs validation loss graph for variant 2

Figure 5.9: training vs validation accuracy and training vs validation loss for ResTraceNet variant 2

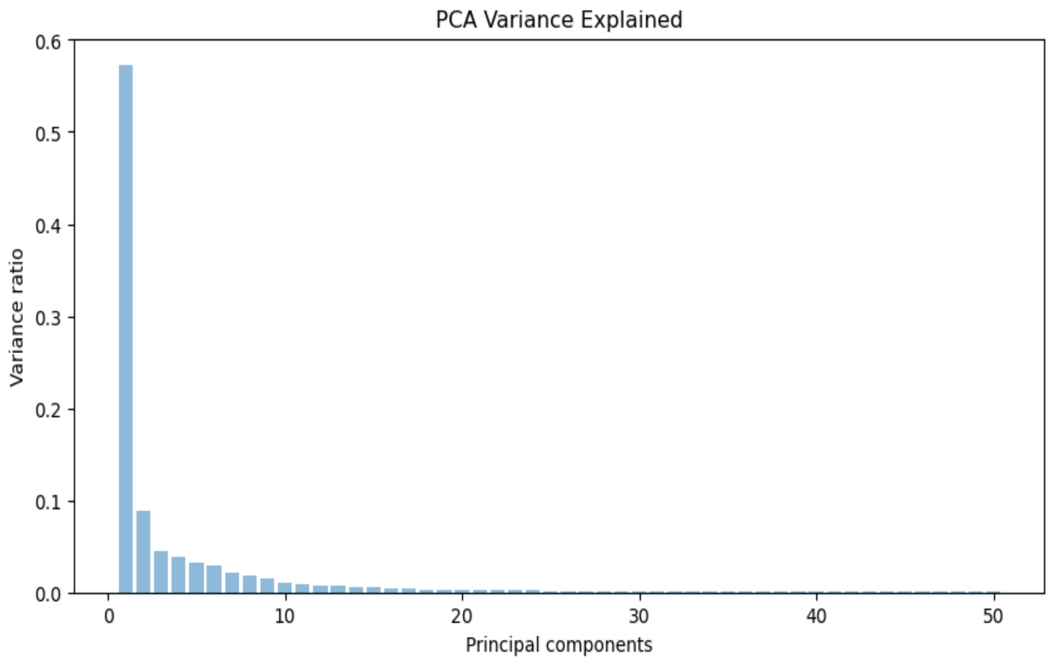
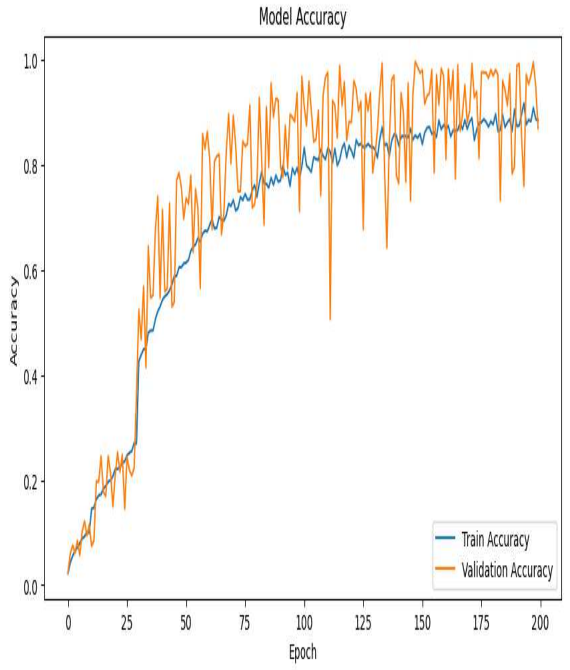
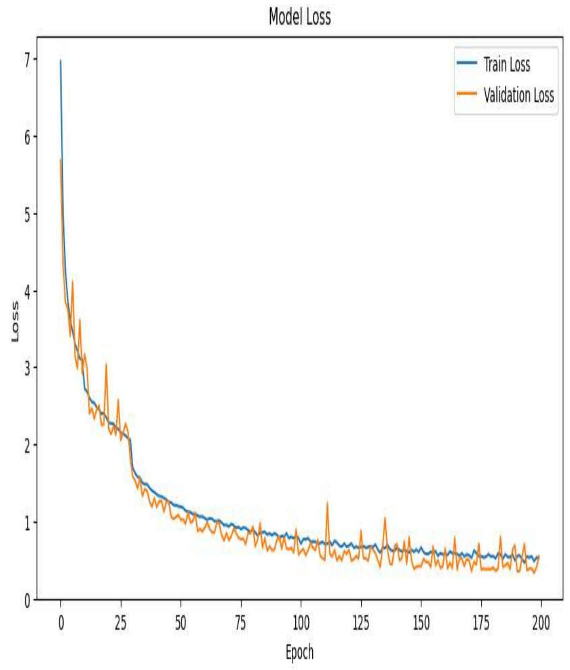


Figure 5.10: The PCA variance explained graph for ResTracenet Variant 3



(a) Training vs validation accuracy graph for ResTraceNet variant 3



(b) Training vs validation loss graph for ResTraceNet variant 3

Figure 5.11: training vs validation accuracy and training vs validation loss for ResTraceNet variant 3

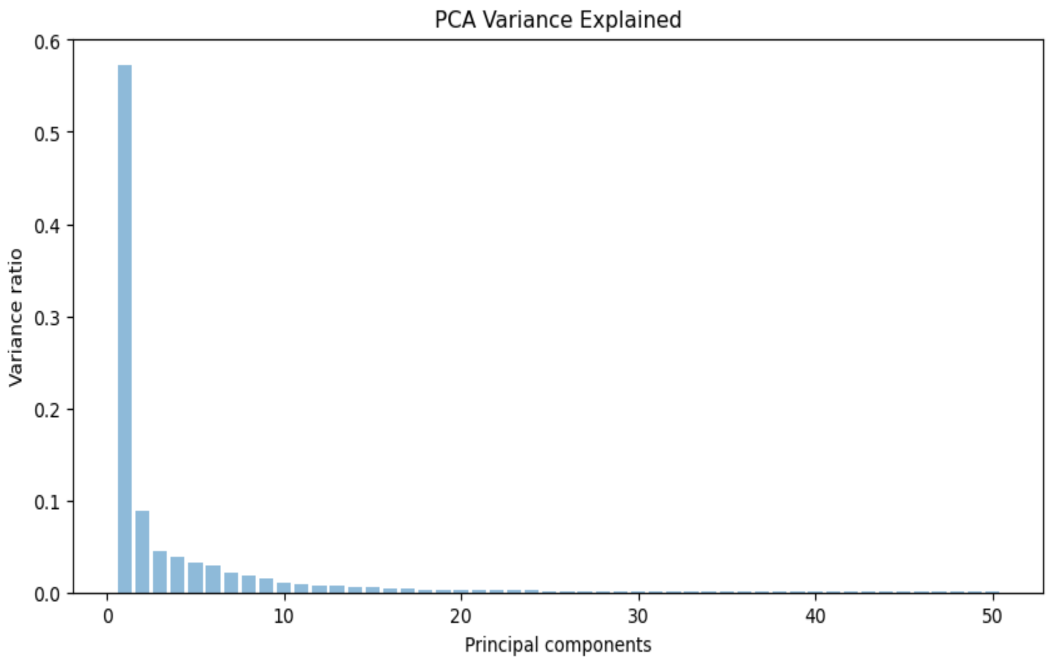
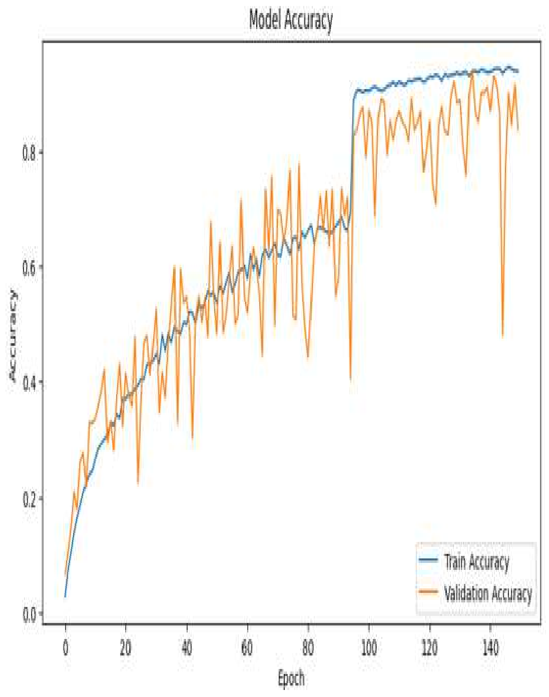
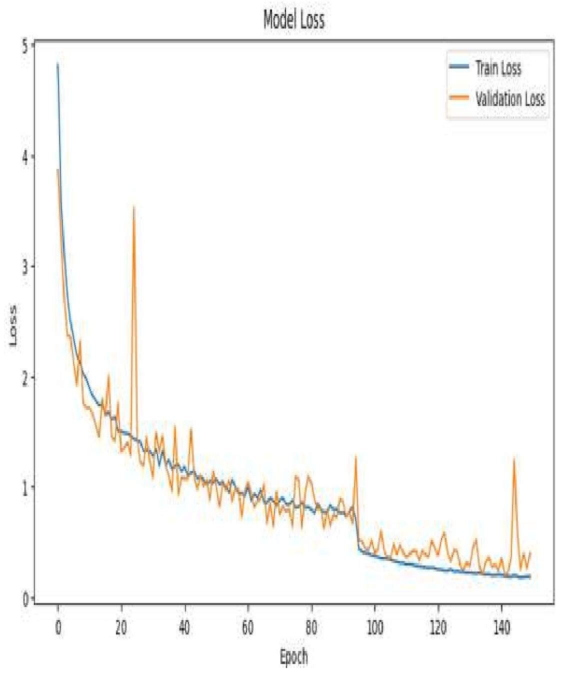


Figure 5.12: The PCA variance explained graph for GRUTrace model



(a) Training vs validation accuracy graph for GRUTrace



(b) training vs validation loss graph for GRUTrace

Figure 5.13: Training vs validation accuracy and training vs validation loss for GRUTrace

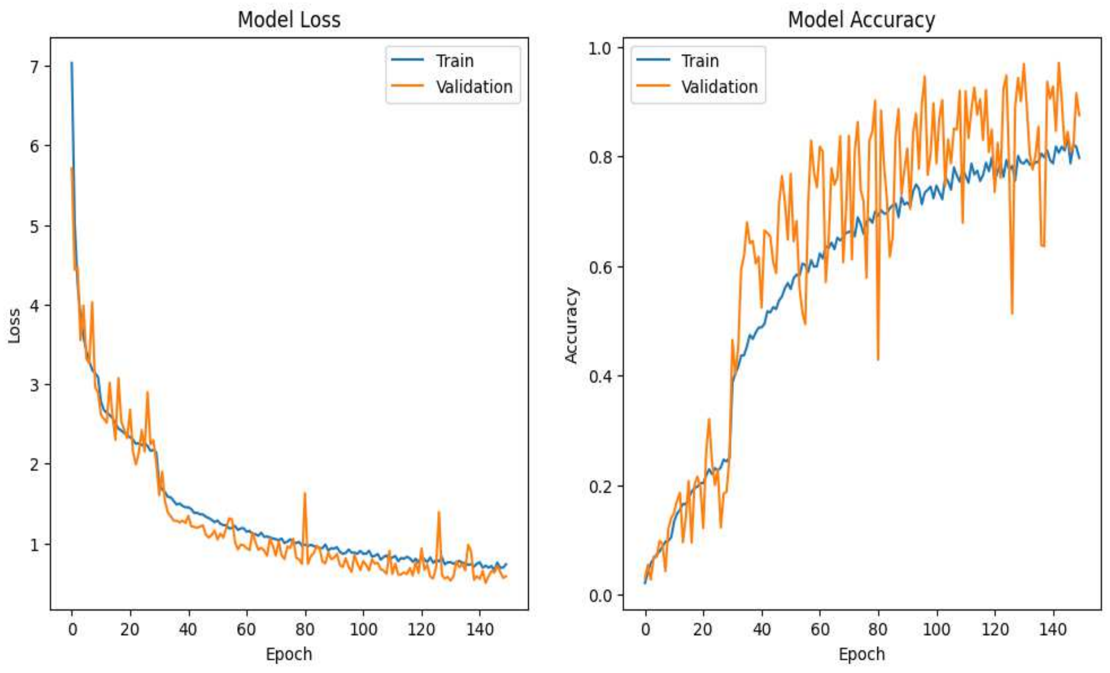
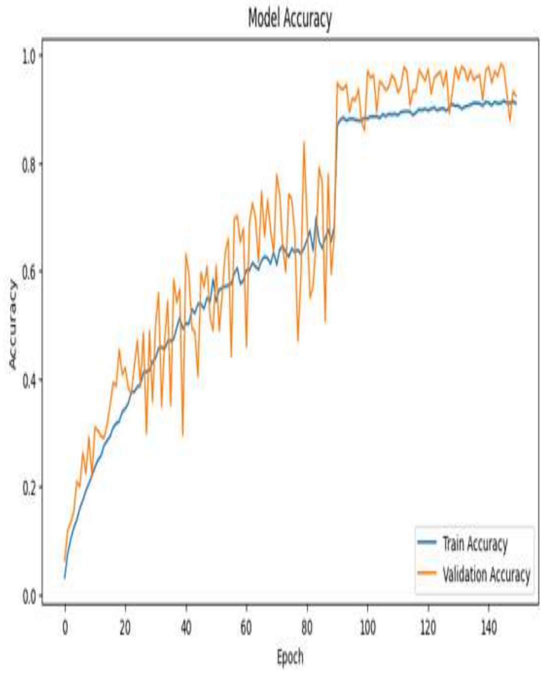
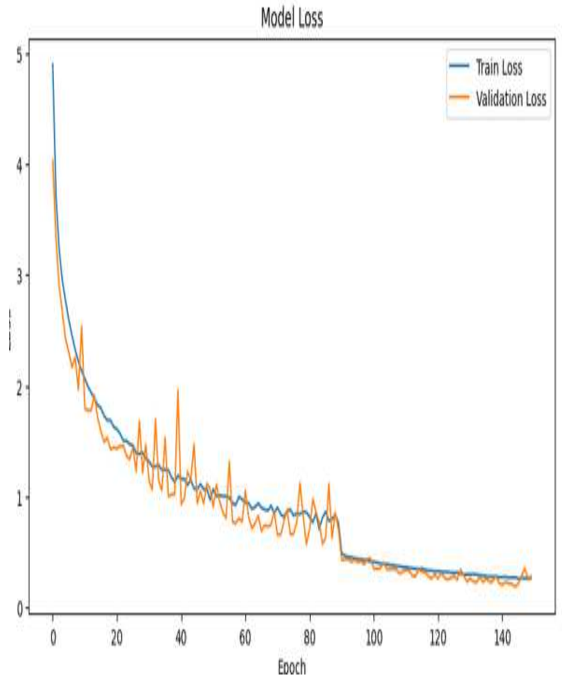


Figure 5.14: The training vs. validation accuracy and loss graph for ResTraceNet model using 100 traces



(a) Training vs validation accuracy graph for GRUTrace using 100 traces



(b) Training vs validation loss graph for GRU-Trace using 100 traces

Figure 5.15: Training vs validation accuracy and training vs validation loss for GRUTrace using 100 traces

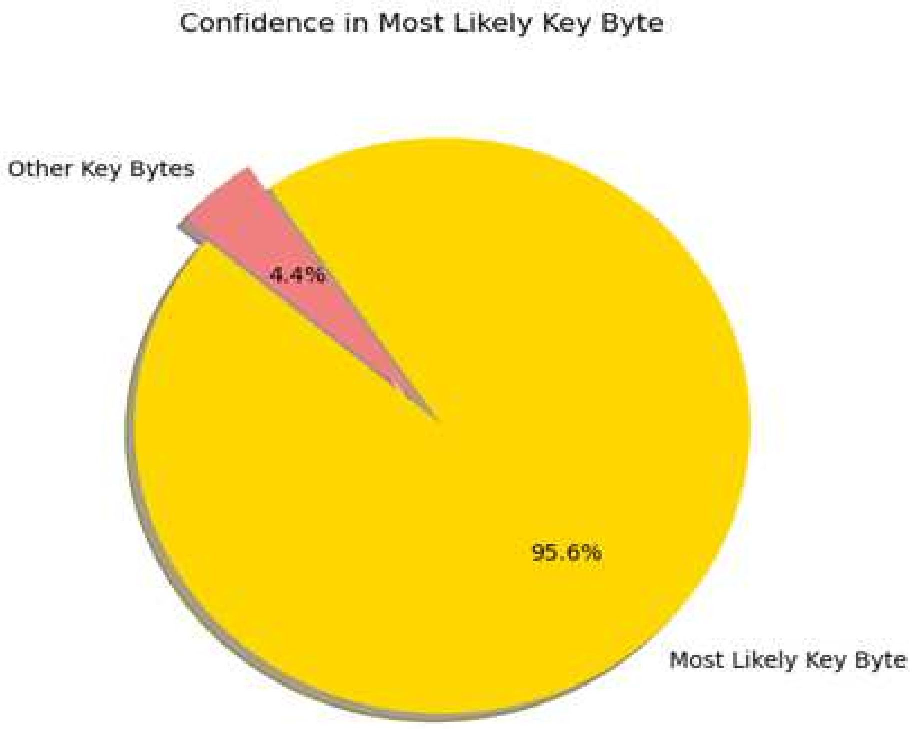


Figure 5.16: Pie chart representing the confidence in the most likely key byte for ResTraceNet model

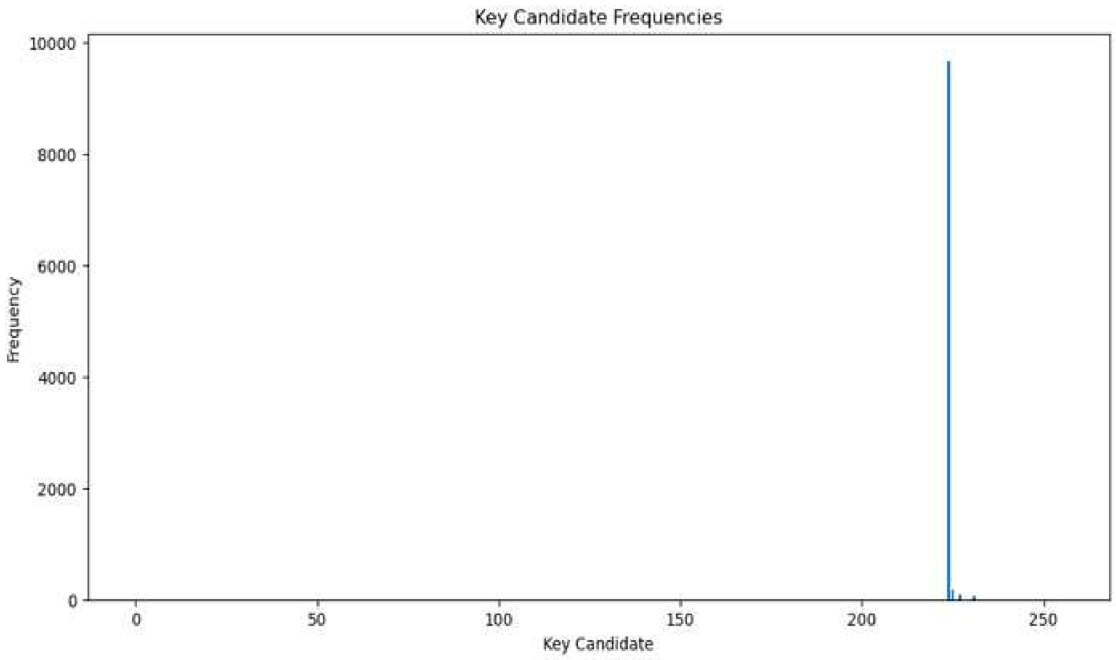


Figure 5.17: Histogram of key candidate frequencies using ResTraceNet model

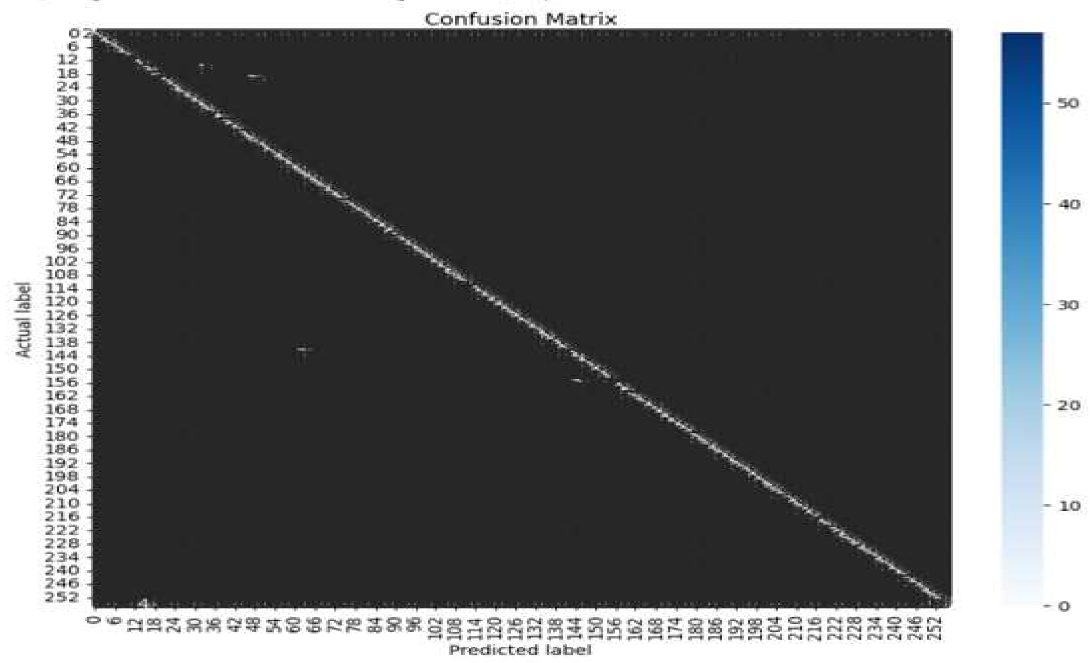


Figure 5.18: confusion matrix for ResTraceNet model using 100 traces

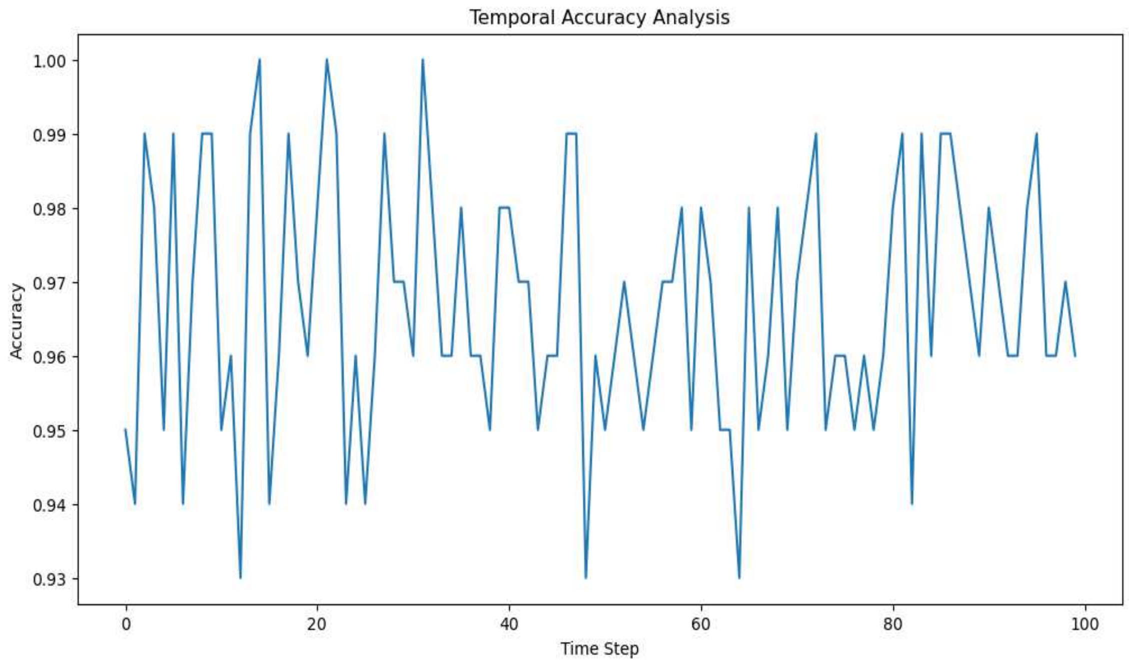


Figure 5.19: Temporal accuracy analysis for ResTraceNet model

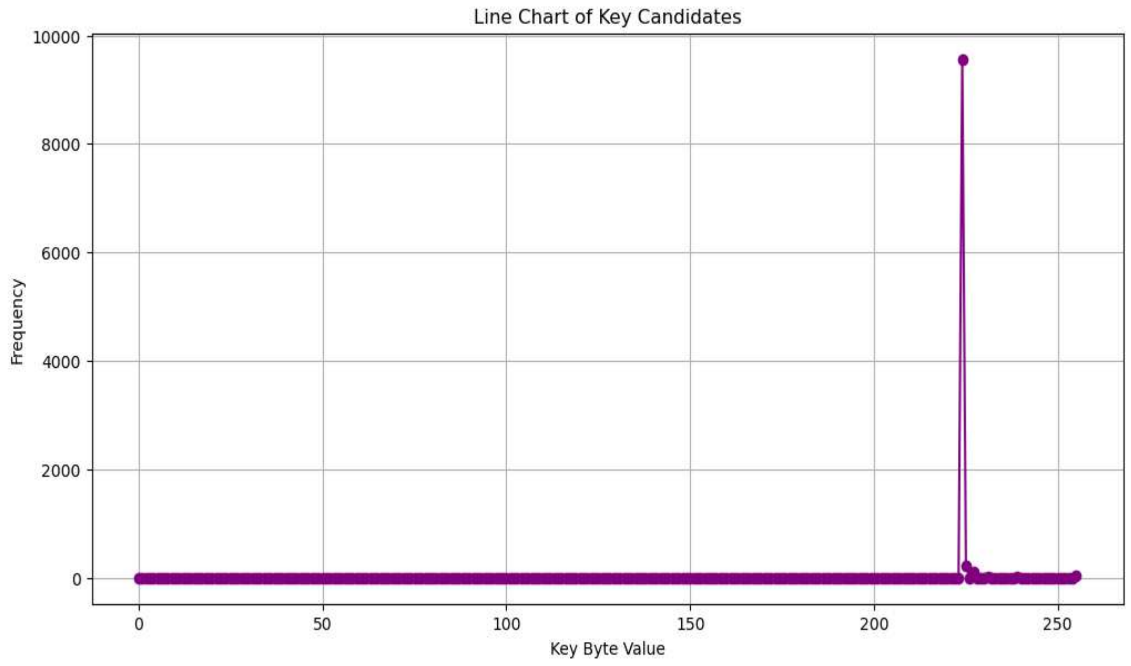


Figure 5.20: Line chart for key candidate frequencies for GRUTrace model

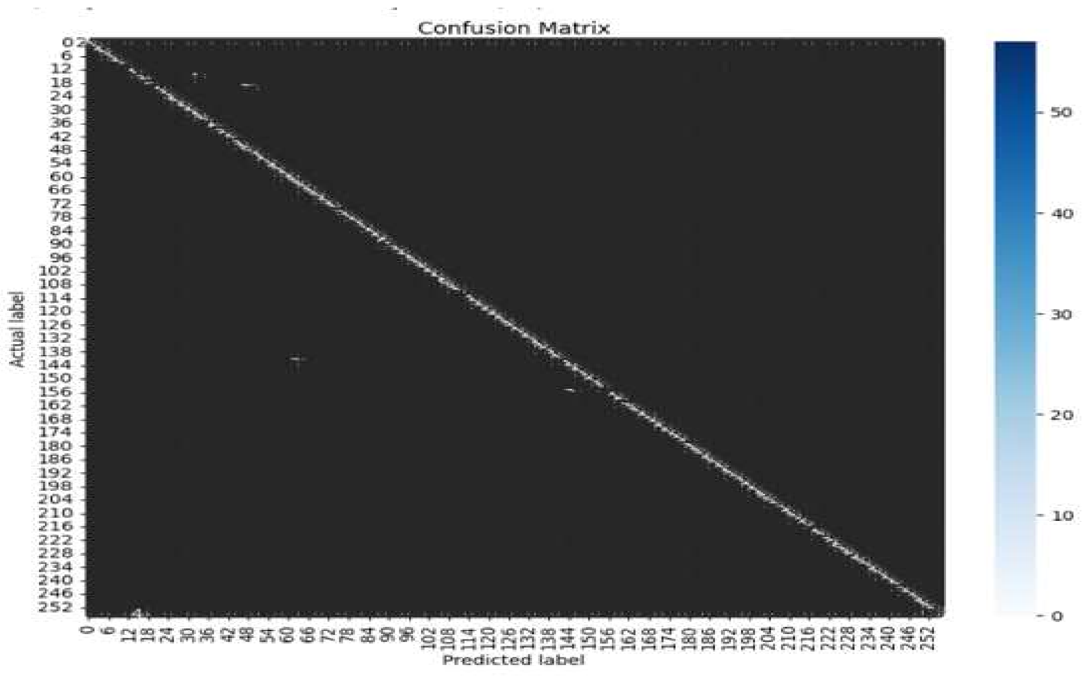
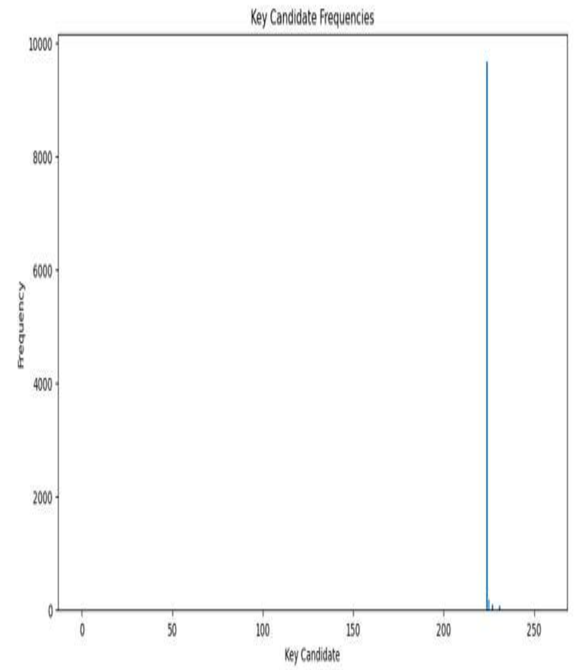
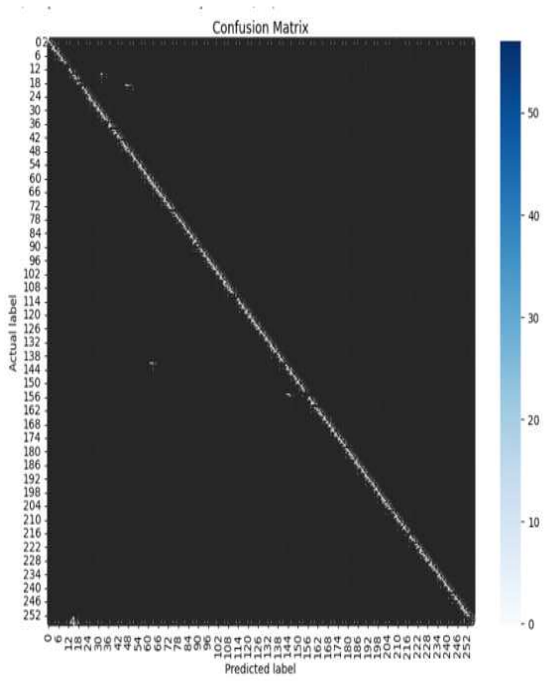


Figure 5.21: Confusion matrix for GRUTrace model



(a) Histogram of key candidate frequency for ResTraceNet and GRUTrace model using 100 power traces



(b) Confusion matrix for ResTraceNet and GRUTrace model using 100 power traces

Figure 5.22: Histogram of key candidate frequency and confusion matrix for GRUTrace and ResTraceNet using 100 power trace

5.4 Comparitive Analysis with existing state of art

Table 5.3: Summary of Experimental Results - Dataset: ASCAD, (Legend: Profiling: Prof., Key Deduction: KD (In Bytes), Not Provided: NP, Accuracy: Acc)

Ref	**Prof.**	**KD & SBox O/P**	**Model**	**Traces**	**Train Acc**	**Val Acc**	**Test Acc**	**KD**
[76]	Yes	Yes	ResNet	240	NP	NP	NP	1
[74]	Yes	Yes	CNN	141/121	NP	NP	NP	1
[79]	Yes	Yes	CODLA	10000	NP	NP	NP	1
[81]	Yes	Yes	Bilinear CNN	8000	NP	NP	NP	1
[105]	Yes	Masked S-box	MLP CNN	NP	NP	NP	81.0% 75.4%	1
Our work	Yes	Yes	ResTraceN	100	82.70%	92.13%	96.80%	1
Our work	Yes	Yes	GRUTrace	100	87.60%	98.70%	98.53%	1

Table 5.3 provides a comparative analysis of key deduction models using the ASCAD dataset, highlighting the differences in methodology and results between our proposed work and existing studies.

- **Paper [76]:** Utilized a ResTraceNet model with 240 traces for profiling, achieving key deduction with S-box output. Training, validation, and testing accuracies were not provided, and the key deduction was 1 byte.

- **Paper [74]:** Used a CNN model with 141/121 traces for profiling. Performed key deduction with S-box output prediction, but specific accuracies were not reported, and the key deduction was 1 byte.

- **Paper [79]:** Leveraged a CODLA model using 10,000 traces for profiling. Key deduction was achieved, but no accuracies were provided, and only 1 byte of the key was deduced.

- **Paper [81]:** Employed a bilinear CNN model with 8,000 traces for profiling. Performed key deduction, but no accuracies were reported. The key deduction was 1 byte.

- **Paper [105]:** Combined MLP/CNN models to profile masked S-box, but specific details about the number of traces, accuracies, and the extent of key deduction were not provided.

- **Proposed Work (ResTraceNet):** Used the ResTraceNet model with 100 traces and achieved a training accuracy of 82.7%, a validation accuracy of 92.13%, and a testing accuracy of 96.8%, successfully deducing 1 key byte.

- **Proposed Work (GRUTrace):** Employed 100 traces with the GRUTrace model, achieving a training accuracy of 87.6%, a validation accuracy of 98.7%, and a testing accuracy of 98.53%, also successfully deducing 1 key byte.

In summary, our proposed work utilized fewer power traces while achieving comparable or better results in key deduction compared to existing methods.

5.5 Summary

In this module, we examined various deep learning methods to enhance the security of the AES algorithm against side-channel attacks, focusing on leveraging ResTraceNet and GRUTrace models to successfully predict the S-box output and deduce the cryptographic key.

Initially, we chose the AES_PTv2 dataset but encountered issues due to the absence of the ciphertext field, which is crucial for making S-box predictions. Consequently, we opted for the ASCAD dataset, which provided robust profiling and attack phase data.

We implemented and evaluated three variants of ResTraceNet. Among these, ResTraceNet Variant 3 demonstrated the most consistent performance, achieving a highest training accuracy of 91.6% and a validation accuracy of 99.4%. The highest testing accuracy recorded for this variant was 96.68%. Additionally, we implemented a GRUTrace model, which attained a training accuracy of 91%, a validation accuracy of 96.8%, and a testing accuracy of 96.28%.

After identifying the most optimal model, we evaluated the models using a subset of 100 power traces to test their efficiency under limited data conditions. The ResTraceNet model achieved training and validation accuracies of 82.7% and 92.13%, respectively, with a testing accuracy of 96.8%. The GRUTrace model, under the same conditions, achieved training and validation accuracies of 87.6% and 98.7%, respectively, and a testing accuracy of 98.53%. In all scenarios, both models successfully deduced the key byte, with the original key byte being [77, 251, 224, 242, 114, 33, 254, 16, 167, 141, 74, 220, 142, 73, 4, 105] and the deduced key byte being 224.

We further validated the effectiveness of these models using various visualizations such as detailed temporal analysis, key candidate distribution, and confidence levels.

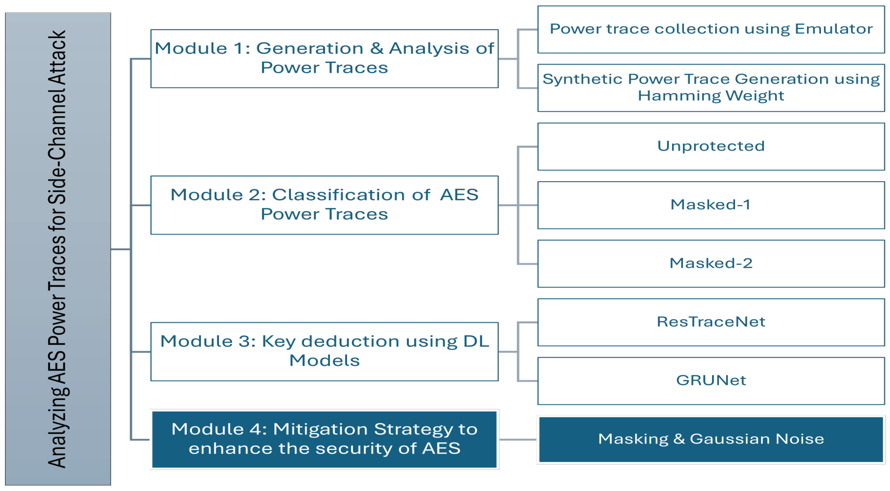
To conclude, the performance of ResTraceNet Variant 3 and the GRUTrace model in accurately predicting the S-box output and deducing the cryptographic key underscores their potential to enhance cryptographic security.

Chapter 6

Module 4: Mitigation strategies

In this module, we explore three mitigation strategies to protect AES implementations against side-channel attacks. The strategies implemented to conceal the power traces include adding Gaussian noise to the power traces, masking the ciphertext using a random mask along with Gaussian noise, and applying masks to both plaintext and ciphertext while also introducing Gaussian noise.

After implementing these countermeasures, we utilized the ResTraceNet model, which demonstrated the highest accuracy in Module 3, to predict the S-Box output and subsequently deduce the key. This approach will determine whether the implemented mitigation strategies effectively obscure the relationship between the power traces and the cryptographic operations, thereby protecting the AES cryptographic implementation against side-channel attacks. Figure 6.1 provides an overview highlighting Module 1 of the research contributions.



Analyzing AES Power Traces for Side-Channel Attack
Module 1: Generation & Analysis of Power Traces
Power trace collection using Emulator
Synthetic Power Trace Generation using Hamming Weight
Module 2: Classification of AES Power Traces
Unprotected
Masked-1
Masked-2
Module 3: Key deduction using DL Models
ResTraceNet
GRUNet
Module 4: Mitigation Strategy to enhance the security of AES
Masking & Gaussian Noise

Figure 6.1: Research Overview: Highlighted Module 4

6.1 Methodology

In this analysis, we employ three unique methods to enhance the security of the AES algorithm by mitigating the effectiveness of side-channel attacks on obtained power traces. Each technique is designed to obscure the power traces and mask the transmitted message, making it difficult for attackers to deduce the relationship between the power traces and cryptographic operations, ultimately preventing the extraction of the cryptographic key. The implemented methods are as follows:

- **Adding Gaussian Noise to the Power Traces:**

  This method involves adding controlled Gaussian noise to the power traces to mask the cryptographic operations. The noise helps to obscure the actual power consumption patterns, making it harder for an attacker to analyze the traces and deduce key information.

- **Masking Ciphertext and Adding Gaussian Noise:**

  In this approach, controlled Gaussian noise is added to the power traces, and random masks are applied to the ciphertext. These masks are derived from a shared key and nonce, ensuring that the information is protected during transmission. This method combines noise addition with ciphertext masking to increase security.

- **Structured Masking and Controlled Noise Addition:**

  This comprehensive method integrates controlled Gaussian noise and structured masking. The plaintext is first masked, then encrypted, and the resulting ciphertext is masked again. This layered approach provides additional security by combining noise addition with both plaintext and ciphertext masking.

Fig. 6.2 shows all the three proposed mitigation strategies.

6.1.1 Importance of shared key and nonce

A shared key and nonce are agreed upon and exchanged between the legitimate sender and receiver. This agreement is crucial as both the key and nonce are necessary

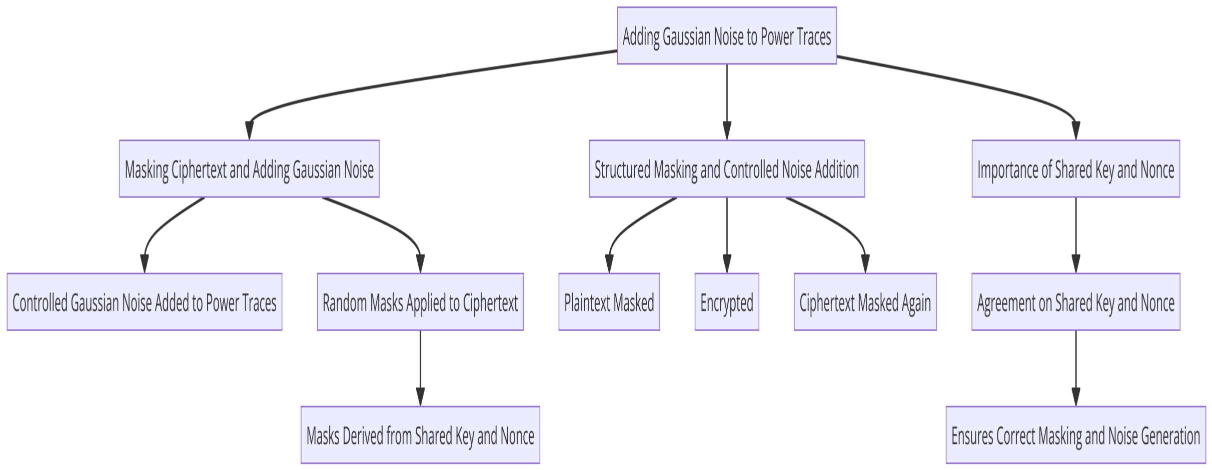

Adding Gaussian Noise to Power Traces
Masking Ciphertext and Adding Gaussian Noise
Structured Masking and Controlled Noise Addition
Importance of Shared Key and Nonce
Controlled Gaussian Noise Added to Power Traces
Random Masks Applied to Ciphertext
Plaintext Masked
Encrypted
Ciphertext Masked Again
Agreement on Shared Key and Nonce
Masks Derived from Shared Key and Nonce
Ensures Correct Masking and Noise Generation

Figure 6.2: All the three proposed mitigation strategies

to derive the seed, ensuring that both parties can generate the same masks and noise sequences independently. This synchronization is essential for correctly masking and unmasking the data without transmitting the masks themselves, thereby enhancing security.

6.1.2 Post-Implementation Analysis

After applying each mitigation strategy, the ResTraceNet model, which demonstrated the highest accuracy in Module 3, is trained to predict the S-Box output and subsequently deduce the cryptographic key. This process evaluates whether the mitigation strategies effectively conceal the relationship between the power traces and cryptographic operations, thereby protecting the AES implementation against side-channel attacks.

6.1.3 Gaussian Noise

Gaussian noise is a statistical noise characterized by a probability distribution function (PDF) that follows a Gaussian or normal distribution. This noise is defined by its mean and variance, and it helps in obscuring the cryptographic operations by altering the power traces [106].

The Gaussian noise formula can be expressed in terms of the PDF as:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{6.1}$$

where:

- x is the noise value.
- μ is the mean of the noise.
- σ^2 is the variance of the noise.
- σ is the standard deviation of the noise.

6.1.4 Adding Gaussian Noise to the Power Trace

In this method, controlled Gaussian noise is added to the power traces. This method aims to mitigate side-channel attacks by making the power traces noisy, thus obscuring the cryptographic operations being performed. The flow of this strategy is as follows:

- **Load the Data:** Power traces, metadata, and labels are loaded from CSV files.
- **Agree on Shared Key and Nonce:** The legitimate sender and receiver agree on a shared key (e.g., '1234567890abcdef') and a nonce ('abcdef1234567890'). This ensures both parties can derive the same seed value independently.
- **Derive Seed:** The seed is obtained by concatenating the shared key and nonce, hashing them using SHA-256, and converting the hash back to an integer value.
- **Generate Gaussian Noise:** Gaussian noise with a mean of 0 and a standard deviation given by the noise factor (0.02) is generated using the seed. This noise is then added to the power traces, making them noisy.
- **Train ResTraceNet Model:** After adding noise to the power traces, the ResTraceNet model is trained to predict the S-Box output and subsequently deduce the cryptographic key. The results are analyzed to determine the effectiveness of the mitigation strategy.

By adding controlled Gaussian noise, the power traces are modified to obscure the cryptographic operations, making it more difficult for an attacker to analyze the traces and extract sensitive information.

$$\hat{y} = \text{ResTraceNet}(P + N(0, \sigma^2) \text{ using } s = \text{int}(\text{SHA-256}(k \parallel n)); \theta) \tag{6.2}$$

Explanation of the Combined Equation

- SHA-256($k \parallel n$): Generates a seed by hashing the concatenation of the key k and nonce n.
- int($\cdot$): Converts the hash to an integer.
- s: Seed value used to generate Gaussian noise.
- $N(0, \sigma^2)$: Gaussian noise with mean 0 and variance σ^2.
- $P + N(0, \sigma^2)$: Adds Gaussian noise to the power traces P.
- ResTraceNet($P+N(0, \sigma^2); \theta$): Trains the ResTraceNet model on the noisy power traces to predict the output $\hat{y}$.
- θ: Parameters of the ResTraceNet model.

6.1.5 Masking ciphertext and adding Gaussian noise

This method aims to enhance the security of cryptographic operations by masking the ciphertext and adding controlled Gaussian noise to the power traces. The objective is to obscure the cryptographic operations, making it more difficult for attackers to deduce sensitive information related to the AES algorithm. The phases for this method are as follows:

- **Load the Dataset:** The dataset, which consists of power traces, metadata, and labels, is loaded.

- **Agree on Shared Key and Nonce:** The legitimate sender and receiver agree upon and share the same shared key and nonce. This ensures both parties can independently derive the same seed value.

- **Derive seed:** The seed is derived using the same principles as in the first strategy. This involves concatenating the shared key and nonce, hashing them with SHA-256, and converting the hash to an integer value.

- **Apply Random Masks to Ciphertext:** Random masks for the ciphertext are generated using the derived seed. The ciphertext is then masked by applying an XOR operation with the generated masks, ensuring its protection and making it harder for attackers to deduce any information.

- **Add Controlled Gaussian Noise:** Controlled Gaussian noise is added to the power traces, further enhancing security.

- **Train ResTraceNet Model:** After masking the ciphertext and adding noise, the ResTraceNet model is trained to predict the S-Box output and ultimately deduce the cryptographic key. This step validates the effectiveness of the implemented strategy in protecting against side-channel attacks (SCAs).

By combining ciphertext masking with the addition of Gaussian noise, this method provides robust protection against SCAs, obscuring cryptographic operations and safeguarding sensitive information. The below equation encapsulates the entire process of masking the ciphertext, adding noise, and using a ResTraceNet model to train and predict the cryptographic key.

$$C_{\text{masked}}, P_{\text{noisy}}, K_{\text{pred}} = \text{ResTraceNet}(P + N(0, \sigma^2), M, L) \tag{6.3}$$

where

$$C_{\text{masked}} = C \oplus \text{PRNG}(\text{int}(\text{SHA-256}(K \parallel N))) \tag{6.4}$$

Explanation of the Combined Equation

- SHA-256$(K \parallel N)$: Generates a seed by hashing the concatenation of the key K and nonce N.

- int(·): Converts the hash to an integer.
- PRNG(·): Generates a pseudorandom mask R using the integer seed.
- $C \oplus R$: Masks the ciphertext C with the pseudorandom mask R.
- $P + N(0, \sigma^2)$: Adds Gaussian noise to the power traces P.
- ReTracesNet(P_{noisy}, M, L): Trains the ResTraceNet model on the noisy power traces, metadata M, and labels L to predict S-Box outputs and deduce the cryptographic key K_{pred}.

6.1.6 Structured Masking and Controlled Noise Addition

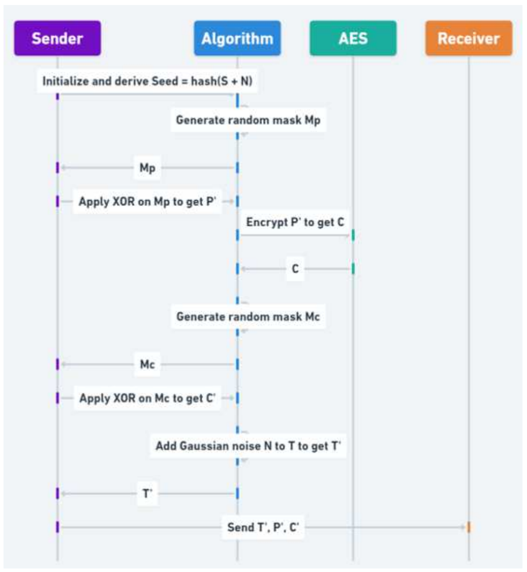
This method presents a comprehensive approach by applying masking to both plaintext and ciphertext and adding controlled Gaussian noise to the power traces. The goal is to obscure all phases of the cryptographic operations in the AES implementation, mitigating the risk of side-channel attacks. The steps involved are as follows:

- **Load the Data:** The dataset, including metadata, power traces, and labels, is loaded.
- **Initialize Shared Key and Nonce:** A shared key and nonce are initialized and agreed upon by the legitimate sender and receiver, ensuring both parties can independently derive the same seed value.
- **Derive Seed:** The seed value is derived by concatenating the shared key and nonce, hashing them using SHA-256, and converting the resulting hash into an integer.
- **Generate Random Masks:**Using the derived seed value, random masks are generated for both plaintext and ciphertext. The plaintext and ciphertext are then masked by applying an XOR operation with these masks. This step is crucial as it ensures that both the input and output of the cryptographic process are obscured, making it difficult for attackers to extract sensitive information.

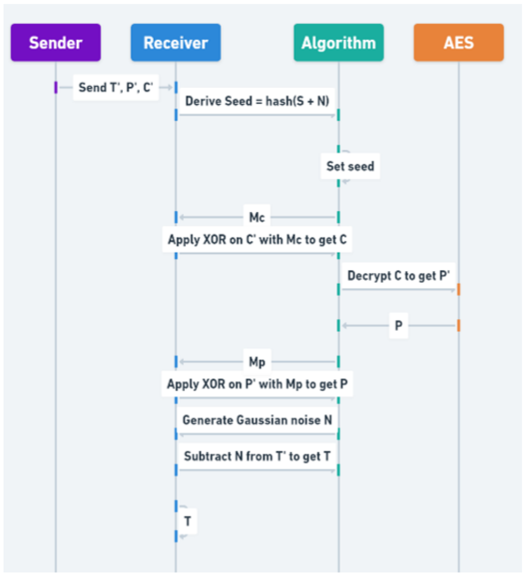
- **Add Controlled Gaussian Noise:** Controlled Gaussian noise is added to the power traces. The noise is generated using the shared key and nonce, ensuring that the traces are consistently masked, which helps prevent attackers from analyzing the power traces.

- **Train ResTraceNet Model:** After applying both the structured masking and controlled noise addition, the ResTraceNet model is trained to predict the S-Box output and subsequently deduce the cryptographic key. This step validates the effectiveness of the comprehensive approach in protecting against side-channel attacks. By combining structured masking with controlled noise addition, this method offers robust protection against side-channel attacks, ensuring that the cryptographic operations are well-protected and sensitive information is secure.

Fig 6.3 shows the sequence diagram for the structured masking and controlled Gaussian noise. As shown in Algorithm 5, the process at the sender side involves using


Sender
Algorithm
AES
Receiver
Initialize and derive Seed = hash(S + N)
Generate random mask Mp
Mp
Apply XOR on Mp to get P'
Encrypt P' to get C
C
Generate random mask Mc
Mc
Apply XOR on Mc to get C'
Add Gaussian noise N to T to get T'
T'
Send T', P', C'

(a) Sequence diagram for structured masking and controlled Gaussian noise at sender side


Sender
Receiver
Algorithm
AES
Send T', P', C'
Derive Seed = hash(S + N)
Set seed
Mc
Apply XOR on C' with Mc to get C
Decrypt C to get P'
P
Mp
Apply XOR on P' with Mp to get P
Generate Gaussian noise N
Subtract N from T' to get T
T

(b) Sequence diagram for structured masking and controlled Gaussian noise at receiver side

Figure 6.3: Sequence diagrams for structured masking and controlled Gaussian noise at both sender and receiver sides

shared keys and nonces to generate masks, applying bitwise XOR for masking, and adding Gaussian noise to power traces.

Algorithm 5 Structured Masking and Controlled Noise Addition at Sender Side

```
Require: Plaintext $P$, AES Key $K$, Power traces $T$, Noise factor $nf$, Shared key $S$, Nonce $N$
Ensure: Noisy traces $T'$, Masked plaintext $P'$, Masked ciphertext $C'$
 1: Initialize: Set the noise factor $= nf$
 2: Use the Shared Key $S$ and Nonce $N$ to derive the random seed: $Seed = \text{hash}(S + N)$
 3: Generate the random mask $M_p$ for the plaintext using the seed:
 4:     np.random.seed(seed)
 5:     $M_p = \text{np.random.randint}(0, 256, \text{size} = P.\text{shape})$
 6: Apply bitwise XOR between plaintext $P$ and masks $M_p$ to get the masked plaintext $P'$:
 7:     $P' = \text{np.bitwise\_xor}(P, M_p)$
 8: Encrypt the masked plaintext with the AES key $K$ to get the ciphertext $C$:
 9:     $C = \text{AES\_encrypt}(P', K)$
10: Generate random mask $M_c$ for the ciphertext using the same seed:
11:     np.random.seed(seed)
12:     $M_c = \text{np.random.randint}(0, 256, \text{size} = P.\text{shape})$
13: Apply bitwise XOR between ciphertext $C$ and masks $M_c$ to get the masked ciphertext $C'$:
14:     $C' = \text{np.bitwise\_xor}(C, M_c)$
15: Add controlled (Gaussian) noise with shape as same as $T$ with mean $= 0$ and standard deviation $= nf$ using the seed:
16:     np.random.seed(seed)
17:     $N = \text{np.random.normal}(0, nf, T.\text{shape})$
18: Add the noise generated $N$ to the traces:
19:     $T' = T + N$
20: Return the noisy traces $T'$, masked plaintext $P'$, and masked ciphertext $C'$
```

As shown in Algorithm 6, the process at the receiver side involves using the same shared keys and nonces to unmask the ciphertext and plaintext, and removing the Gaussian noise from the power traces.

Algorithm 6 Structured Masking and Controlled Noise Addition at Receiver Side

```
Require: Masked ciphertext $C'$, Noisy traces $T'$, Shared key $S$, Nonce $N$, AES Key $K$, Noise factor $nf$
Ensure: Original plaintext $P$, Power traces $T$
 1: Derive the seed using the agreed shared key $S$ and nonce $N$
 2:     $Seed = \text{hash}(S + N)$
 3: Set the seed for the purpose of reproducibility
 4:     np.random.seed(seed)
 5: Unmask the ciphertext $C$ by generating the same random mask $M_c$ for the ciphertext
 6:     $M_c = \text{np.random.randint}(0, 256, \text{size} = C'.\text{shape})$
 7:     $C = \text{np.bitwise\_xor}(C', M_c)$
 8: Decrypt the ciphertext with the AES key $K$ to get the plaintext $P$
 9:     $P' = \text{AES\_decrypt}(C, K)$
10: Unmask the plaintext $P$ by generating the same random mask $M_p$ for the plaintext
11:     $M_p = \text{np.random.randint}(0, 256, \text{size} = P'.\text{shape})$
12:     $P = \text{np.bitwise\_xor}(P', M_p)$
13: Generate the same Gaussian noise with shape as $T$ with mean = 0 and standard deviation = $nf$
14:     $N = \text{np.random.normal}(0, nf, T.\text{shape})$
15: Subtract the noise $N$ from the received noisy power traces $T'$
16:     $T = T' - N$
17: Return the original plaintext $P$ and power traces $T$
```

Mathematical Functions in Mitigation strategy

1. Structured Masking

$$P' = P \oplus M_p \tag{6.5}$$

$$C' = C \oplus M_c \tag{6.6}$$

- P: Original plaintext

- M_p: Random mask for PT
- P': Masked plaintext
- C: Original ciphertext
- M_c: Random mask for CT
- C': Masked ciphertext

2. Controlled Noise Addition

$$T' = T + N \quad \text{where} \quad N \sim \mathcal{N}(0, \sigma^2) \tag{6.7}$$

- T: Original power traces
- N: Gaussian noise with mean = 0 and variance σ^2
- T': Noisy power traces

Reference to Mathematical Functions

As shown in the Mathematical Functions in Mitigation strategy section, the structured masking and controlled noise addition methods are used for securing data.

6.2 Results and discussion

We computed the training vs. validation accuracy and training vs. validation loss graphs for the ResTraceNet model using all three proposed methods and evaluated the output.

6.2.1 Graph Analyis

Adding Gaussian Noise to the Power Traces

Fig. 6.4 demonstrate the training and validation accuracy and loss for Method 1. The accuracy graph shows an increase in both training and validation accuracies across the epochs. However, there is significant fluctuation in the validation accuracy, indicating some instability and overfitting. The training vs. validation loss graph shows a steady decline, but the validation loss remains higher and more variable than the training loss.

In conclusion, Method 1 has failed to effectively protect the AES implementation against side-channel attacks, as it does not sufficiently obscure the cryptographic operations and patterns that could lead to the deduction of sensitive information.

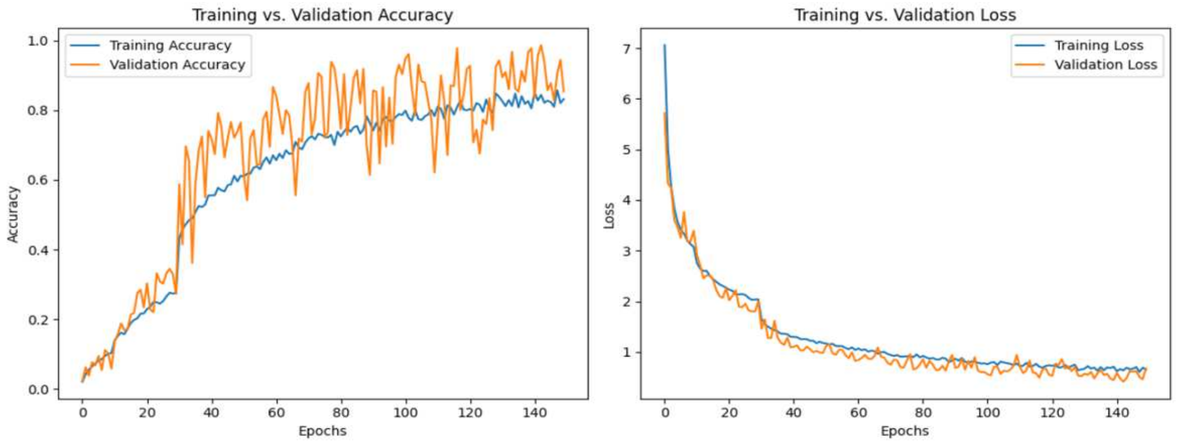


Figure 6.4: Training vs validation accuracy and training vs validation loss graph for method 1

Masking Ciphertext and Adding Gaussian Noise

Fig. 6.5 show the training vs. validation accuracy and training vs. validation loss for Method 2. Similar to Method 1, there is an increasing trend in both training and validation accuracy. However, the validation accuracy shows significant fluctuation, and the validation loss is considerably high with some variations.

We can conclude that Method 2 has also failed to counter side-channel attacks effectively, despite masking the ciphertext, which added a new layer of security. This is evidenced by the unstable validation plot.

Structured Masking and Controlled Noise Addition

Figures 6.6 display the training vs. validation accuracy and loss for this comprehensive approach. Compared to the previous methods, Method 3 shows very poor performance, with the model attaining low accuracy and high loss values. The graphs clearly demonstrate that the model has failed to learn and generalize, indicated by the high validation loss and poor validation accuracy.

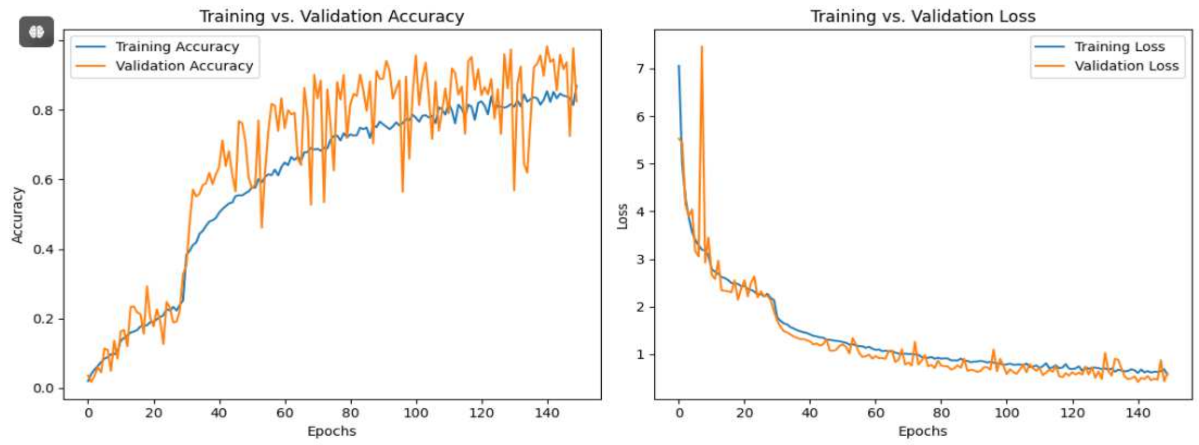


Figure 6.5: Training vs validation accuracy and training vs validation loss graph for method 2

This comprehensive approach of adding two layers of security, along with Gaussian noise, successfully combats side-channel attacks. This is evidenced by the weak performance of the ResTraceNet model, indicating that this approach effectively obscures the latent relationships between the power traces and cryptographic operations. As shown by the model's inability to learn and predict the S-Box output and the cryptographic key, this method has proven successful in mitigating side-channel attacks.

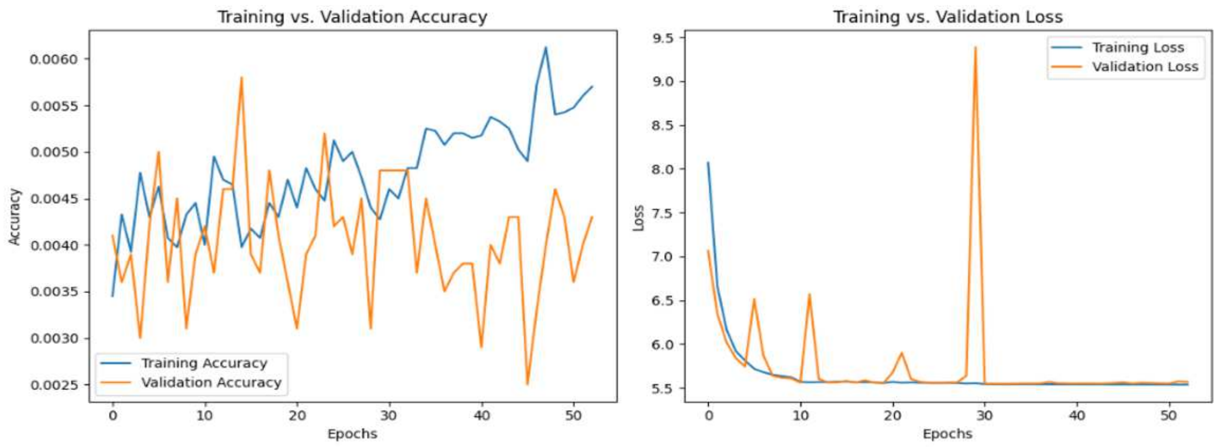


Figure 6.6: Training vs validation accuracy and training vs validation loss graph for method 3

6.3 Comparitive analysis with existing state of art

The Table6.1 provides a comparative analysis of key deduction models and mitigation strategies for AES side-channel attacks using the ASCAD dataset, highlighting the methodologies and results of our proposed work against existing studies.

Table 6.1: Comparative analysis of Module 4 with existing work

Paper	**Dataset**	**Attack type**	**Key deduction and S-box output**	**Model Used**	**Mitigation**
[72]	ASCAD (publicly available)	Profiling	Yes	Deep Learning, CNN, MLP	No
[76]	ASCAD (publicly available)	Profiling	Yes	Deep 2-D CNN, CWT, Scalograms, GoogLeNet, InceptionV3, VGG16, MobileNetV2	No
[77]	ASCAD (publicly available)	Profiling	Yes	Multilabel Classification, CNN, MLP Ensemble Learning	No
[87]	ASCAD	Profiling	Yes	Transfer Learning, CNN, MLP, ResNet, AutoEncoder, Pre-training, Freezing, Fine-tuning	No
[107]	ASCAD	Profiling	Yes	ResNet	No
Proposed work	**ASCAD**	**Profiling**	**Yes**	**ResTraceNet**	**Yes**

- **Papers [72],[76],[77],[87] and [107]:** These studies used the publicly available ASCAD dataset for both S-box prediction and key deduction. They employed deep learning models such as CNN, MLP, and more sophisticated methods like transfer learning and ensemble learning. However, none of these studies implemented mitigation strategies to combat side-channel attacks (SCAs) in AES implementations.

- **Proposed Work:** Our study also utilized the ASCAD dataset and focused on profiling for key deduction and S-box output prediction using advanced deep learning models such as ResTraceNet and GRUTrace. Unlike previous works, we incorporated mitigation strategies to obscure the power traces and protect cryptographic implementations against SCAs, thereby enhancing overall security.

This comparative analysis highlights the unique contributions of our work in integrating both key deduction and effective mitigation techniques using advanced deep learning models.

6.4 Summary

In this module, we examined and evaluated three mitigation techniques to protect the AES algorithm from SCAs. Each method aimed to obscure the power traces and cryptographic operations, making it difficult for the model to identify correlations and deduce sensitive information. The methods explored were:

- Adding Gaussian Noise to the Power Traces

  In this method, we introduced controlled Gaussian noise to the power traces to mask the cryptographic operations. Although it provided some obfuscation, it was unable to stabilize the model performance effectively against SCAs, as demonstrated by the fluctuations in validation accuracy and higher validation loss.

- Masking Ciphertext and Adding Gaussian Noise

  This technique added another layer of security by masking the ciphertext, derived from a seed value using a shared key and nonce, along with Gaussian noise. Despite the extra security measure, this method resulted in an unstable validation plot, indicating that it did not provide sufficient protection against side-channel attacks.

- Structured Masking and Controlled Noise Addition

  This comprehensive method combined Gaussian noise addition with structured masking of both plaintext and ciphertext. It proved to be the most effective approach. By masking both the input and output and adding noise, it successfully obscured the cryptographic operations. This effectiveness was indicated by the poor performance of the ResTraceNet model, which showed high validation loss and poor validation accuracy.

After implementing all three strategies, we used the ResTraceNet model, which had previously shown excellent performance in predicting the S-Box and deriving the key in Module 3, to evaluate our proposed methods. The results clearly show that Method 3 was successful in countering side-channel attacks by effectively obscuring the relationship between cryptographic operations and power traces in an AES implementation.

In summary, structured masking and controlled noise addition emerged as the most robust mitigation strategy, significantly enhancing the security of the AES algorithm and its cryptographic operations.

Chapter 7

Conclusions

Fig. 7.1 provides a comprehensive overview of our methodology and results in enhancing the security of the AES algorithm against SCAs using DL models. It begins with the power trace generation phase, employing two methods, and proceeds to the implementation of various deep learning techniques to deduce the cryptographic key. It also highlights the mitigation strategies implemented to combat SCAs. The diagram can be explained as follows:

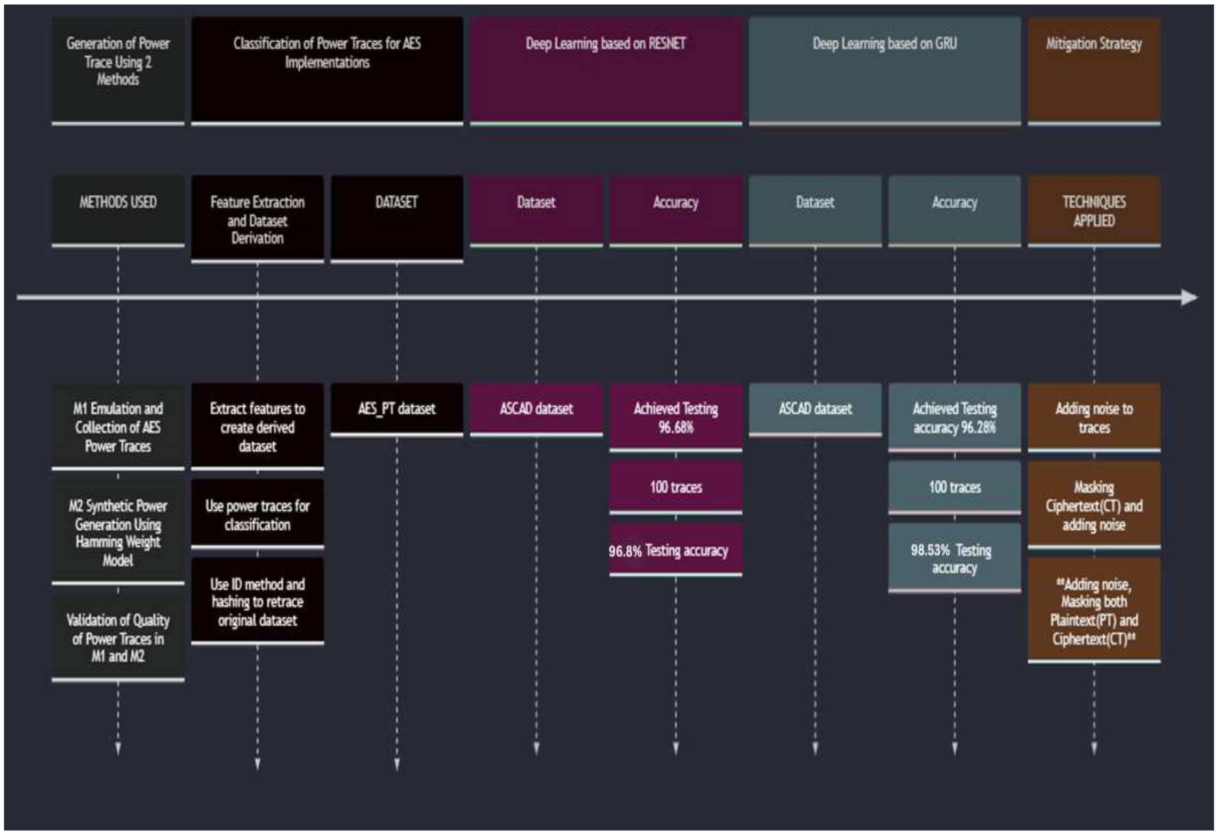

Generation of Power Trace Using 2 Methods
Classification of Power Traces for AES Implementations
Deep Learning based on RESNET
Deep Learning based on GRU
Mitigation Strategy
METHODS USED
Feature Extraction and Dataset Derivation
DATASET
Dataset
Accuracy
Dataset
Accuracy
TECHNIQUES APPLIED
M1 Emulation and Collection of AES Power Traces
M2 Synthetic Power Generation Using Hamming Weight Model
Validation of Quality of Power Traces in M1 and M2
Extract features to create derived dataset
Use power traces for classification
Use ID method and hashing to retrace original dataset
AES_PT dataset
ASCAD dataset
Achieved Testing 96.68%
100 traces
96.8% Testing accuracy
ASCAD dataset
Achieved Testing accuracy 96.28%
100 traces
98.53% Testing accuracy
Adding noise to traces
Masking Ciphertext(CT) and adding noise
Adding noise, Masking both Plaintext(PT) and Ciphertext(CT)

Figure 7.1: The summary of contribution depicting methodology and results

7.1 Contributions

7.1.1 Generation of AES Power Traces

- **M1 Emulation:** This involves the collection of power traces using actual hardware simulation.
- **M2 Synthetic Generation:** Utilizes the concept of Hamming weight to generate synthetic AES power traces.
- **Validation:** Ensures the quality and alignment of the collected power traces.

7.1.2 Classification of Power Traces Using AES Implementation:

- **Dataset used:** ASCAD Dataset which cntains EM and power traces.
- **Feature Extraction:** Extracts relevant features using various techniques to curate a derived dataset.
- **Classification:** Uses these features to classify the power traces into various AES implementations.
- **ID Method and Hashing:** Retraces the original power traces to validate data integrity.

7.1.3 Deep Learning Based on ResTraceNet:

- **Dataset Used:** AES_PT v2.
- **Accuracy:** Achieved a testing accuracy of 96.68%.
- **100 Traces:** Using 100 power traces, achieved a testing accuracy of 96.8%.

7.1.4 Deep Learning Based on GRUTrace

- **Dataset:** ASCAD dataset.
- **Accuracy:** Achieved a testing accuracy of 96.28%.

- **100 Traces:** Using only 100 power traces, the model achieved a testing accuracy of 98.53%.

7.1.5 Mitigation Strategies

- **Method 1:** Adding noise to the power traces.
- **Method 2:** Adding noise to the power traces and masking ciphertext.
- **Method 3:** Adding noise to the power traces, masking plaintext, and ciphertext to provide an enhanced level of security.

This summary encapsulates the detailed methodology, dataset choices, model implementations, and mitigation strategies employed to counter side-channel attacks.

7.1.6 Module summary

- **Module 1:**

  In this module, we collected power traces for a masked AES implementation from both real and synthetic sources. Actual power traces were gathered using the Syscomp CGR-101 oscilloscope, while synthetic traces were programmatically generated using the HW concept. We performed detailed timing analysis and used various techniques to align and correlate both sets of power traces to validate the synthetic traces. We achieved a high correlation of 0.986 between the synthetic and real power traces using Random Forest regression analysis, indicating that Random Forest provided the best alignment and correlation.

- **Module 2:**

  We applied various feature extraction techniques to improve the classification accuracy of three AES implementations. Methods such as permutation importance, recursive feature elimination, L1 regularization, mutual information, and SHAP were utilized. We employed PCA to identify the most important statistical properties—mean, minimum, maximum, and standard deviation—and leveraged a support vector machine to classify the power traces into unprotected, masked1, and masked2 implementations, yielding high accuracy. We also retraced the original power traces from the derived dataset using hashing and the ID method, providing traceability.

- **Module 3:**

  his module explored two powerful deep learning models, ResTraceNet and GRUTrace, to predict the S-box output and deduce the key. We opted for the ASCAD dataset due to the absence of an integral field, ciphertext, in the AES_PT dataset. Among the three ResTraceNet variants employed, Variant 3 exhibited the highest training accuracy of 91.6%, validation accuracy of 99.4%, and testing accuracy of 96.68%. The GRUTrace model achieved training and validation accuracies of 91% and 96.8%, respectively, with a testing accuracy of 96.28%. Using 100 power traces, ResTraceNet achieved a testing accuracy of 96.8%, while GRUTrace achieved 98.53%, successfully deducing the key byte in all scenarios. Since the relationship between the S-box output, plaintext, and key byte was based on the third key byte, we were able to deduce the third key byte, which was "224".

- **Module 4:**

  We focused on three mitigation strategies to protect AES implementation from side-channel attacks: adding Gaussian noise to power traces, masking ciphertext and adding Gaussian noise, and finally, structured masking and controlled noise addition. The most effective method was the structured masking and controlled noise addition, which showed poor performance of the best variant ResTraceNet model, indicating that it efficiently obscured the relationship between the power traces and cryptographic operations.

Chapter 8

Discussion

This section will discuss about the limitations and potential future work of our research.

8.1 Limitations

Although we achieved significant and promising results, we encountered several limitations during our study.

- **Dataset Constraints:** Initially, we used the AES_PTv2 dataset, but it lacked the ciphertext field, which is integral for S-box prediction and key deduction. This constraint forced us to switch to the ASCAD dataset, highlighting our dependency on the availability of comprehensive datasets for effective training and validation.
- **Synthetic Data Generation:** Achieving high correlation and fidelity in the generated synthetic traces remains challenging. The synthetic traces might not fully capture all the complexities and inherent noise of real-world data, which can affect the accuracy of the analysis.
- **Model Overfitting:** The initial variants of the ResTraceNet model exhibited signs of overfitting, as indicated by fluctuations in validation accuracy and loss. This underscores the need for more robust regularization techniques or advanced methods to enhance model generalization.
- **Computational Resources:** Training deep learning models such as ResTraceNet and GRUTrace requires substantial computational power and time, especially when dealing with large datasets and multiple epochs. This resource-intensive process can limit the feasibility of extensive experimentation and optimization in environments with limited computational resources.

- **Mitigation Strategies:** While employing structured masking and controlled noise addition, the complexity of implementation and potential performance overhead must be considered. Ensuring practical applicability without significantly disrupting the system's performance remains a challenge.

Addressing these limitations in future research will be crucial for developing more robust and generalized solutions to enhance cryptographic security against side-channel attacks.

8.2 Future work

- **Improved Dataset Collection and Diversity:** Future research should focus on exploring a wider range of datasets for side-channel analysis, including various encryption algorithms and their hardware implementations. This will help create more generalized models capable of effectively countering side-channel attacks across diverse environments.

- **Enhanced Synthetic Data Generation:** Advanced techniques need to be developed to create synthetic data that accurately represents raw power traces. This involves incorporating more sophisticated noise models and hardware-specific properties to enhance the fidelity of the synthetic traces.

- **Developing More Powerful Deep Learning Models:** Exploring hybrid models that combine CNNs, RNNs, and transformers could significantly improve performance. Feature extraction methods like wavelet packet decomposition and higher-order statistical features should be employed to identify more latent patterns and relationships. Automating feature extraction using AI-based selection methods can streamline and enhance model efficacy.

- **Using Advanced Regularization Techniques:** Implementing robust regularization techniques such as dropout variations, weight decay, and data augmentation can help resolve overfitting issues, leading to better model generalization.

- **Hyperparameter Tuning:** Greater emphasis should be placed on hyperparameter tuning to optimize the performance of deep learning models. Automating hyperparameter tuning and using exhaustive search techniques can help identify the most optimal model configurations for the given datasets.

- **Design and Develop Mitigation Strategies:** There should be a stronger focus on designing and testing new mitigation strategies to protect cryptographic implementations against a range of side-channel attacks.

- **Efficient Computational Models:** For real-world applications, it is imperative to reduce the computational burden of training deep learning models. Research should delve into more efficient algorithms and hardware accelerations such as Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs) to speed up the training process.

Bibliography

[1] P. A. Chandrakasan, *Series on Integrated Circuits and Systems.* Springer.

[2] J. D. S. G. S. S. A. R. Anupam Golder, Debayan Das, "Practical approaches toward deep-learning-based cross-device power side-channel attack," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. Issue Number, 2019.

[3] P. C. v. O. Frank Piessens, "Side-channel attacks: A short tour," *IEEE Security Privacy*, vol. Volume Number, 2024.

[4] P. J.-J. Quisquater, "Side channel attacks," Tech. Rep. CRYPTREC EX-1047-2002, Cryptographic Technology and Evaluation Committee, 2002.

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

[6] M. Alioto, M. Poli, and S. Rocchi, "Power analysis attacks to cryptographic circuits: a comparative analysis of dpa and cpa," in *2008 International Conference on Microelectronics*, pp. 333–336, 2008.

[7] N. Hanley, M. O'Neill, M. Tunstall, and W. P. Marnane, "Empirical evaluation of multi-device profiling side-channel attacks," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, 2014.

[8] X. Lu, C. Zhang, and D. Gu, "Attention - based non-profiled side-channel attack," in *2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1–6, 2021.

[9] F.-X. Standaert, "Introduction to side-channel attacks," in *Secure Integrated Circuits and Systems*, pp. 27–42, Springer, 2010.

[10] A. Rădulescu and M. O. Choudary, "Side-channel attacks on masked bitsliced implementations of aes," *Cryptography*, vol. 6, no. 3, p. 31, 2022.

[11] M. Yasin, B. Mazumdar, S. S. Ali, and O. Sinanoglu, "Security analysis of logic encryption against the most effective side-channel attack: Dpa," in *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, pp. 97–102, Oct 2015.

[12] W. J. B. D. C. Owen Lo, "Power analysis attacks on the aes-128 s-box using differential power analysis (dpa) and correlation power analysis (cpa)," *Journal of Cyber Security Technology*, vol. 1, 2016.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] H. Maghrebi, "Deep learning based side channel attacks in practice." Cryptology ePrint Archive, Paper 2019/578, 2019. https://eprint.iacr.org/2019/578.

[15] B. K. Jefferson, MO, *Convolutional Neural Networks with Swift for Tensorflow*. New York, NY: Springer, 2021.

[16] G. Shen, Q. Tan, H. Zhang, P. Zeng, and J. Xu, "Deep learning with gated recurrent unit networks for financial sequence predictions," *Procedia Computer Science*, vol. 131, pp. 895–903, 2018. Recent Advancement in Information and Communication Technology:.

[17] "Understanding the hamming weight metric." https://math.stackexchange.com/questions/2405535/understanding-the-hamming-weight-metric#:~:text=Definition%3A%20Hamming%20weight&text=The%20hamming%20weight%20of%20b,ni%3D0zi., 2024. Accessed: 2024-07-11.

[18] E. Bursztein, "Hacker guide to deep learning side-channel attacks: The theory." https://elie.net/blog/security/hacker-guide-to-deep-learning-side-channel-attacks-the-theory, 2020. Accessed: 2024-07-11.

[19] H. Gupta, S. Mondal, R. Majumdar, N. S. Ghosh, S. Suvra Khan, N. E. Kwanyu, and V. P. Mishra, "Impact of side channel attack in information security," in *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, pp. 291–295, Dec 2019.

[20] M. Tehranipoor, N. N. Anandakumar, and F. Farahmandi, "Power analysis attacks on aes," in *Hardware Security Training, Hands-on!*, pp. 137–161, Springer, 2023.

[21] N. Tada, Y.-i. Hayashi, T. Mizuki, and H. Sone, "An efficient analysis method of the electromagnetic emissions in the frequency domain," in *2012 Proceedings of SICE Annual Conference (SICE)*, pp. 68–72, Aug 2012.

[22] S. Sharma, H. Gupta, and S. K. Sharma, "An overview of acoustic side channel attack," *International Journal of Computer Science and Communication Networks*, vol. 3, no. 1, pp. 27–31, 2013.

[23] G. S. B. K. Puneet Vashisht, Munish Rattan, "A survey of defensive mechanisms for cognitive radio networks," *ACM Computing Surveys*, vol. 52, no. 4, pp. 1–35, 2019.

[24] T. Kim and Y. Shin, "Thermalbleed: A practical thermal side-channel attack," *IEEE Access*, vol. 10, pp. 25718–25731, 2022.

[25] N. Hanley, M. O'Neill, M. Tunstall, and W. P. Marnane, "Empirical evaluation of multi-device profiling side-channel attacks," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, 2014.

[26] L. Wu, L. Weissbart, M. Krček, H. Li, G. Perin, L. Batina, and S. Picek, "On the attack evaluation and the generalization ability in profiling side-channel analysis." Cryptology ePrint Archive, Paper 2020/899.

[27] B. Timon, "Non-profiled deep learning-based side-channel attacks." Cryptology ePrint Archive, Paper 2018/196, 2018. `https://eprint.iacr.org/2018/196`.

[28] A. Ali, S. Khan, and S. Ullah, "Security and privacy challenges in iot-based wireless sensor networks: A comprehensive survey," *IET Smart Cities*, vol. 5, no. 2, pp. 142–158, 2023.

[29] D. Kwon, S. Hong, and H. Kim, "Optimizing implementations of non-profiled deep learning-based side-channel attacks," *IEEE Access*, vol. 10, pp. 5957–5967, 2022.

[30] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, pp. 433–451. Springer, 2007.

[31] S. Sicari, A. Rizzardi, L. A. Griepentrog, and A. Coen-Porisini, *Security and Privacy in Internet of Things (IoT): Models, Algorithms, and Implementations*, pp. 181–199. Springer, 2016.

[32] R. E. Smith and M. R. Patel, "Asynchronous cryptographic system: Side channel attacks through differential power analysis (dpa)," in *Proceedings of the 9th Annual IEEE Symposium on Applications and the Internet (SAINT'08)*, pp. 215–218, IEEE, 2008.

[33] S. Ali, J. Zhou, X. Zhang, and Z. Wang, *Optimal Power Allocation and Outage Analysis for Multi-Relay NOMA Systems Using Dinkelbach Method*, pp. 173–186. Springer, 2016.

[34] M. S. a. T. F. c. Takaya Kubota a, Kota Yoshida b, "Deep learning side-channel attack against hardware implementations of aes," in *ScienceDirect*, vol. 87, 2021.

[35] N. Sklavos and X. Zhang, *Wireless Security and Cryptography: Specifications and Implementations.* Boca Raton, FL: CRC Press, 2007.

[36] A. Tripathy and B. Singh, "A study of aes software implementation for iot systems," in *2022 3rd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pp. 1–4, 2022.

[37] A. Abdullah, “Advanced encryption standard (aes) algorithm to encrypt and decrypt data.” `https://www.researchgate.net/publication/317615794_Advanced_Encryption_Standard_AES_Algorithm_to_Encrypt_and_Decrypt_Data`, 2017. Accessed: 2024-06-11.

[38] X. Z. Jian Zhou and S. Ali, “Optimal power allocation and outage analysis for multi-relay noma systems using dinkelbach method,” in *Advances in Communication, Signal Processing, and VLSI*, pp. 81–99, Springer, 2016.

[39] A. A. Pammu, K.-S. Chong, W.-G. Ho, and B.-H. Gwee, “Interceptive side channel attack on aes-128 wireless communications for iot applications,” in *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 650–653, 2016.

[40] A. Name, “The cost of power density: A quantitative model for power density requirements and costs of data centers,” in *Computer Science and its Applications*, pp. 315–326, Springer, 2016.

[41] A. Name, “An overview of acoustic side channel attack,” *International Journal of Computer Network and Information Security*, vol. 6, no. 1, pp. 62–67, 2014.

[42] M. M. Malikg, “A hierarchy of limitations in machine learning,”

[43] C. Janiesch, P. Zschech, and K. Heinrich, “Machine learning and deep learning,” *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021.

[44] M. A. Koosha Sharifani, “Machine learning and deep learning: A review of methods and applications,” vol. 10, pp. 3897–3904, World Information Technology and Engineering Journal,, 2023.

[45] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, “A survey of deep learning and its applications: A new paradigm to machine learning,” *Archives of Computational Methods in Engineering*, vol. 27, no. 4, pp. 1071–1092, 2020.

[46] P. Z. . K. H. Christian Janiesch, “Machine learning and deep learning,” *Electronic Markets*, vol. 31, no. 3, pp. 663–677, 2021.

[47] R. Maheswari and M. Krishnamurthy, “Profiling and non-profiling key retrieval attacks in programmable object interfaces using deep learning cryptanalytic techniques: A survey,” in *2024 2nd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)*, pp. 1–5, 2024.

[48] A. Aljuffri, C. Reinbrecht, S. Hamdioui, and M. Taouil, “Impact of data preprocessing techniques on deep learning based power attacks,” in *2021 16th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, pp. 1–6, 2021.

[49] J. M. Rodriguez, J. Garcia, and M. Lopez, "A dual approach for ensuring privacy in public cloud storage," in *The Seventh International Conference on Performance, Safety and Robustness in Complex Systems and Applications (PESARO 2017)*, pp. 20–25, IARIA, 2017.

[50] Z. Wu, C. Shen, and A. van den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[52] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[53] "Dropout in neural networks." `https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9?gi=65d63be57c65`. Accessed: 2024-06-11.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[55] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[56] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[57] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, 2018.

[58] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[59] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[60] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[61] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[62] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

[63] I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” *arXiv preprint arXiv:1608.03983*, 2016.

[64] P. Socha, J. Brejník, and M. Bartik, “Attacking aes implementations using correlation power analysis on zybo zynq-7000 soc board,” in *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–4, 2018.

[65] Q. Hu, X. Fan, and Q. Zhang, “An effective differential power attack method for advanced encryption standard,” in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 58–61, 2019.

[66] A. T. Mozipo and J. M. Acken, “Power side channel attack of aes fpga implementation with experimental results using full keys,” in *2021 IEEE International Conference on Design Test of Integrated Micro Nano-Systems (DTS)*, pp. 1–6, 2021.

[67] S. Darbar, M. J., and D. Selvakumar, “Side channel leakage assessment strategy on attack resistant aes architectures,” in *2020 24th International Symposium on VLSI Design and Test (VDAT)*, pp. 1–6, 2020.

[68] L. Lathrop, “Differential power analysis attacks on different implementations of AES with the ChipWhisperer nano.” Cryptology ePrint Archive, Paper 2020/1008, 2020. `https://eprint.iacr.org/2020/1008`.

[69] S. D. Putra, A. D. W. Sumari, I. Asrowardi, E. Subyantoro, and L. M. Zagi, “First-round and last-round power analysis attack against aes devices,” in *2020 International Conference on Information Technology Systems and Innovation (ICITSI)*, pp. 410–415, 2020.

[70] M. Masoumi, “Novel hybrid cmos/memristor implementation of the aes algorithm robust against differential power analysis attack,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 7, pp. 1314–1318, 2020.

[71] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, “Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 3, pp. 148–179, 2019.

[72] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, “Deep learning for side-channel analysis and introduction to ascad database,” *Journal of Cryptographic Engineering*, vol. 10, pp. 163–188, 2020.

[73] J. Zhang, M. Zheng, J. Nan, H. Hu, and N. Yu, "A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, pp. 73–96, 2020.

[74] J. Liu, S. Zheng, and L. Gu, "Effective deep learning-based side-channel analyses against ascad," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 514–523, 2021.

[75] Z. Luo, M. Zheng, P. Wang, M. Jin, J. Zhang, and H. Hu, "Towards strengthening deep learning-based side channel attacks with mixup," *arXiv preprint arXiv:2103.05833*, 2021.

[76] A. Garg and N. Karimian, "Leveraging deep cnn and transfer learning for side-channel attack," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 91–96, 2021.

[77] L. Zhang, X. Xing, J. Fan, Z. Wang, and S. Wang, "Multilabel deep learning-based side-channel attack," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1207–1216, 2021.

[78] X. Huang, M. M. Wong, A. T. Do, and W. L. Goh, "A backpropagation extreme learning machine approach to fast training neural network-based side-channel attack," in *2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1–6, 2021.

[79] J. Chen, J.-S. Ng, N. A. Kyaw, N. K. Z. Lwin, K.-S. Chong, Z. Lin, J. S. Chang, and B.-H. Gwee, "Non-profiling based correlation optimization deep learning analysis," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2246–2250, 2022.

[80] D. Kwon, S. Hong, and H. Kim, "Optimizing implementations of non-profiled deep learning-based side-channel attacks," *IEEE Access*, vol. 10, pp. 5957–5967, 2022.

[81] P. Cao, C. Zhang, X. Lu, D. Gu, and S. Xu, "Improving deep learning based second-order side-channel analysis with bilinear cnn," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3863–3876, 2022.

[82] L. Wu, G. Perin, and S. Picek, "The best of two worlds: Deep learning-assisted template attack." Cryptology ePrint Archive, Paper 2021/959, 2021. `https://eprint.iacr.org/2021/959`.

[83] S. Paguada, L. Batina, I. Buhan, and I. Armendariz, “Playing with blocks: Toward re-usable deep learning models for side-channel profiled attacks,” *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2835–2847, 2022.

[84] Y. Li, W. Zhuang, L. Yu, and Z. Qin, “Study of cnn and lstm based on ascad database with different kinds of noise,” in *2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*, pp. 375–379, 2022.

[85] J. H. Daehyeon Bae, Jongbae Hwang, “Deep learning-based attacks on masked aes implementation,” *Journal of Internet Technology*, vol. 23, pp. 897–902, 2022.

[86] G. Perin, L. Wu, and S. Picek, “AISY - deep learning-based framework for side-channel analysis,” *IACR Cryptology ePrint Archive*, vol. 2021, p. 357, 2021.

[87] M. Fang, B. Mao, and W. Hu, “A transfer learning approach for electromagnetic side-channel attack and evaluation,” in *2022 7th International Conference on Integrated Circuits and Microsystems (ICICM)*, pp. 636–640, 2022.

[88] D. Bae, D. Park, G. Kim, M. Choi, N. Lee, H. Kim, and S. Hong, “Autoscaled-wavelet convolutional layer for deep learning-based side-channel analysis,” *IEEE Access*, vol. 11, pp. 95381–95395, 2023.

[89] B. Jocelyn and F. Clementine, “Power analysis attacks and countermeasures for elliptic curve cryptosystems,” in *Proceedings of the 2005 Cryptographic Hardware and Embedded Systems Workshop (CHES 2005)*, pp. 1–15, Springer, 2005.

[90] Saelig, “Pspc019 - product.” `https://www.saelig.com/product/PSPC019.htm`. Accessed: 2024-06-11.

[91] M. Nakanose, Y. Kodera, T. Kusaka, and Y. Nogami, “Consideration of the side-channel attack to speck implemented on arduino uno,” in *2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW)*, pp. 339–345, 2021.

[92] P. Duhamel and M. Vetterli, “Fast fourier transforms: A tutorial review and a state of the art,” *Signal Processing*, vol. 19, no. 4, pp. 259–299, 1990.

[93] B. A. Draper, K. Baek, M. S. Bartlett, and J. Beveridge, “Recognizing faces with pca and ica,” *Computer Vision and Image Understanding*, vol. 91, no. 1, pp. 115–137, 2003. Special Issue on Face Recognition.

[94] Y. Chen, “A new methodology of spatial cross-correlation analysis,” *PLOS ONE*, vol. 10, no. 5, p. e0126158, 2015.

[95] M. Gholizadeh, J.-P. Chiles, and J. P. M. Syvitski, "Geostatistical modelling of palaeochannel sedimentology: a 3d approach," *Mathematical Geosciences*, 2022.

[96] W. Yuchi, E. Gombojav, B. Boldbaatar, J. Galsuren, S. Enkhmaa, B. Beejin, G. Naidan, C. Ochir, B. Legtseg, T. Byambaa, P. Barn, S. B. Henderson, C. R. Janes, B. P. Lanphear, L. C. McCandless, T. K. Takaro, S. A. Venners, G. M. Webster, and R. W. Allen, "Evaluation of random forest regression and multiple linear regression for predicting indoor fine particulate matter concentrations in a highly polluted city," *Environmental Pollution*, vol. 245, pp. 746–753, 2019.

[97] Urioja, "Aesptv2." `https://github.com/urioja/AESPTv2`, 2024. Accessed: 2024-06-11.

[98] Y. Yang, J. Li, and Y. Yang, "The research of the fast svm classifier method," in *2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 121–124, 2015.

[99] B. Lepri, N. Oliver, E. Letouze, A. S. Pentland, and P. Vinck, "Ai and big data: The birth of a new intelligence," in *Privacy and Power in the Digital Age*, pp. 89–105, Springer, 2020.

[100] T. Schultz, S. Brady, and U. Mueller, "A mycophagous insect fungus mutualism: Ants that cultivate yeasts of the genus escovopsis," *BMC Genetics*, vol. 19, no. 1, p. 33, 2018.

[101] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, "Statistical mechanics of complex networks," *Physical Review E*, vol. 69, no. 6, p. 066138, 2004.

[102] U. Rioja, L. Batina, J. L. Flores, and I. Armendariz, "Auto-tune pois: Estimation of distribution algorithms for efficient side-channel analysis," *Computer Networks*, vol. 198, p. 108405, 2021.

[103] U. Rioja, L. Batina, I. Armendariz, and J. L. Flores, "Towards human dependency elimination: Ai approach to sca robustness assessment," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3906–3921, 2022.

[104] I. H. Sarker *et al.*, "A review on artificial intelligence in cybersecurity: New vulnerabilities and security intelligence," *Journal of Cryptographic Engineering*, vol. 13, no. 3, pp. 321–345, 2023.

[105] E. Cagli, C. Dumas, and E. Prouff, "Breaking a masked aes implementation using a deep learning-based attack," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*, pp. 15–27, Association for Computing Machinery, 2021.

[106] "Gaussian white noise." https://www.sciencedirect.com/topics/computer-science/gaussian-white-noise. Accessed: 2024-06-11.

[107] S. Sun, H. Yu, A. Wang, C. Wei, *et al.*, "Dual-path hybrid residual network for profiled side-channel analysis," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. PP, no. 99, pp. 1–1, 2024.