DESIGN AND IMPLEMENTATION OF A PORTABLE HARDWARE DEVICE FOR ANALYZE SIDE CHANNEL ATTACKS

Karunarathna Mudiyanselage Chethanike Anjana Nimsara

IT21285974

Bsc (Hons) Degree In Information Technology
Specializing In Cyber Security

Department Of Computer System Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

DECLARATION

I declare that this is my own work, and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as article or books).

Signature: Date: 09/04/2025

Signature of the Supervisor: Date: 09/04/2025

ABSTRACT

Side-channel attacks (SCAs) have emerged as a successful threat to cryptographic security by exploiting unintentional information leakage in physical hardware, i.e., power consumption, to reveal sensitive information. This paper details a complete, embedded system for conducting power-based side channel analysis of the Data Encryption Standard (DES) encryption algorithm by leveraging real time data acquisition, machine learning-based inference, and autonomous output visualization.

Hardware implementation uses microcontrollers like Arduino Nano and ESP32 and INA219 current sensor to measure voltage, current, power, and energy consumed when performing cryptographic functions. Real-time measured data is saved on an SD card and controlled remotely via Bluetooth with the Serial Bluetooth Terminal application. The project is ported to an advanced ESP32 based platform to compensate for memory and processing constraints with a view to its onboard Bluetooth capability and ease of integration with sensors.

Statistical features of the power traces are employed to train machine learning models namely Multilayer Perceptrons (MLPs) to predict sensitive cryptographic parameters like plaintext and key values. The model achieved very good performance in terms of accuracy, like 84% accuracy in predicting plaintext across 1000 training epochs. The trained model was also deployed on an embedded platform for real-time inference and the output shown on a 16x2 LCD screen, showcasing end to end automation of the pipeline from data collection to attack execution.

This work not only validates the feasibility of mounting side channel attacks on DES with inexpensive portable equipment but also demonstrates the feasibility in practice of applying machine learning to automate the evaluation of cryptographic weaknesses. The system constructed herein provides a scalable, low-cost platform for hardware security research, education and testing, as well as a pointer to the need for better countermeasures for embedded cryptographic implementations.

ACKNOWLEDGEMENT

First and foremost I would like to thank my supervisor Mr. Kavinga Yapa for his ongoing guidance, encouragement, and valuable insights throughout the duration of this research. His experience and constructive criticism shaped the direction of this project and encouraged me to consider novel approaches in the arena of side channel analysis.

I would also like to take this chance to extend my heartfelt thanks to my co-supervisor Mr. Amila Senarathna whose technical contribution and advice were instrumental in the successful implementation and analysis phases of this project. His analytical explanations and real world experience made me grasp embedded systems and machine learning applications much more easily.

I am also grateful to all of the faculty members and technical personnel of our department for providing the facilities and academic atmosphere necessary to complete this work.

A special note of thanks to my family and friends who provided me with constant encouragement and moral support throughout this journey. They kept me going through the tough times because they had faith in me. Lastly I would like to thank all the individuals who contributed directly or indirectly towards the success of this project. This project could not have been a success without the assistance of this wonderful group of people.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviations	Full Term
AES	Advanced Encryption Standard
CPU	Central Processing Unit
CSV	Comma-Separated Values
DES	Data Encryption Standard
EM	Electromagnetic
ESP32	Espressif Systems 32-bit Microcontroller
GPIO	General-Purpose Input/Output
GUI	Graphical User Interface
IDE	Integrated Development Environment
IoT	Internet of Things
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
RNG	Random Number Generator
SCA	Side-Channel Attack
SVM	Support Vector Machine
VCC	Voltage Common Collector (Power Supply Pin)
RSA	Rivest–Shamir–Adleman
SSH	Secure Shell

INTRODUCTION

Overview

Most of the securities systems they often Focused on most of the software security side or the software level. So as an example some of the exploits like Trojan horses and ransomware I'll consider as a big threat for the organizations and companies but even though there are hardware based attacks are present. Those kinds of attacks are called side channel attacks so basically they get information Without the acknowledgement of the authorized hardware device. So inside channel attacks they directly steal information from a hardware device and after a particular analyzing and identify the vulnerabilities of the particular system and exploit. [1] The leaking information may be electromagnetic radiation, power consumption, visible light, execution time and sound. By analyzing these different information we can mount a side channel attack so as an example by analyzing the sound we can perform acoustic side channel attack [2] and also by analyzing the power consumption during a particular cryptographic operation or cryptographic encryption we can find the plain text or cryptography keys like a master keys during encryption process. So there are two types of analysis attacks simple power analysis and differential power analysis.

In simple power analysis It carried out the attack by examined current consumption over a period of time. [3] By using this method it can identify different power profiles or What type of function is perform during the particular time. The differential power analysis method uses statistical method to discern subtle differences in power consumption that correlate with specific data operations. [4] These kind of attacks are more challenge for IoT devices or embedded systems and other hardware implementations like smart cards they are power consumption can be leveraged to break cryptographic keys

And this project is focused on symmetric encryption algorithms Based cryptographic operations and see how strong encryption algorithms which are used in organizational companies. Our objective is to conduct power analysis based on attack and understand vulnerabilities. This study is especially pertinent given the present state of security issues, where a greater demand for strong hardware-level cryptography protection has arisen due to the proliferation of connected devices.

Background Literature

Side-channel attacks (SCAs) represent a serious threat to cryptographic implementations taking advantage of physical leakages of hardware devices instead of depending on weaknesses of cryptographic algorithms alone. Power-based analysis has been the most explored side channel attack with its viability and relative ease of implementation. They exploit the correlation between power consumed and internal data or data processed of a cryptographic device enabling an attacker to deduce secret data like cryptographic keys. The roots of power analysis as a severe threat go back to the seminal work of Paul Kocher and others in 1999 that presented both Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [5]. These methods demonstrated that even secure encryption algorithms such as DES and RSA can be broken using non-intrusive power consumption measurements.

Simple Power Analysis (SPA) is when the power traces of a device are directly observed and patterns or anomalies representing particular operations or data values are determined. This method relies on visual inspection or pattern matching and may be employed to reveal hidden key bits, control flow, or conditional jumps. However SPA may be successful only typically if the signal to noise ratio is high and the operations of interest are observable in the power traces. Differential Power Analysis (DPA) introduced a more potent statistical approach however. By gathering a large number of power traces for varying inputs and using statistical analysis like correlation or difference of means, DPA allows secret keys to be extracted even under noisy conditions. This technique has been generalized and analyzed in the literature extensively including the extension to higher order DPA attacks against masking countermeasures [6].

Much research has been devoted to breaking down and enhancing DPA methods. For example Brier et al. (2004) introduced Correlation Power Analysis (CPA), which relates hypothetical intermediate values (derived by applying known plaintexts and an estimated key) to actual power consumption data. CPA is now one of the most popular types of DPA because it is simple and effective. CPA takes a power model typically the Hamming weight or Hamming distance model as a relation between data values and power consumption. The effectiveness of our CPA relies on the extent to which the power model describes the physical reality of the device and so additional work on precise modeling of power consumption is needed. [7]

Utilization of hardware platforms for SCA is also crucial. While initial attacks were demonstrated on smart cards and embedded systems the method has since been used on a wide range of platforms from microcontrollers and FPGAs to general purpose CPUs. Recent years have seen the proliferation of low cost experimentation platforms based on microcontrollers like Arduino and ESP32 enabling students and researchers to perform real world attacks using very minimal equipment. They have gained popularity in academia and education environments because of ease of use and the availability of a large number of open source tools. For instance O'Flynn's ChipWhisperer platform offers an end to end environment for side channel research from hardware to firmware to analysis software. [8]

Several works have investigated the susceptibility of cryptosystems like AES, DES and RSA to power based side-channel attacks. DES has been of particular interest because it has a well-structured and predictable implementation. Researchers have shown the possibility of DPA and CPA attacks in extracting DES keys from a small number of traces. [6] In one such study Mangard examined the S-box and key schedule processing in DES for areas of interest in the power trace where the leakage is most prominent. The study highlighted the requirement for very precise trace and synchronization alignment particularly in software-implemented systems where the execution timing is not deterministic. Thus most of the existing side-channel analysis frameworks incorporate the pre-processing steps of trace alignment, noise removal, and feature reduction through mechanisms such as Principal Component Analysis (PCA) [6].

To counter the increasing threat of SCAs various countermeasures have been suggested and investigated. These can be classified into masking, hiding and randomization techniques. Masking consists of inserting random values (masks) into intermediate computations thus decorrelating the data from the power traces. Though effective in theory. Masking can be broken by higher-order attacks that exploit several leakage points. Hiding methods try to decrease the signal to noise ratio by power balancing among operations through balancing, current flattening or dummy operation insertion. Randomization mechanisms like random instruction insertion and shuffling and render power traces less predictable between encryptions and making statistical attacks weaker. Each of these countermeasures has trade offs in terms of performance, complexity and effectiveness and a combination is generally required for good protection. [9]

More recently there have been applications of machine learning techniques to power based SCAs with great success. Researchers have used neural networks, support vector machines and random forests to model and take advantage of intricate patterns in power traces not modeled by conventional statistical methods. These techniques have a tendency to outperform conventional methods when confronted with noisy or partially misaligned traces. Deep learning methods in the form of CNNs were demonstrated to be capable of high key recovery with few traces and minimal pre processing which has questioned the effectiveness of traditional countermeasures. [10] Machine learning attacks, nevertheless usually need vast amounts of training data and are prone to overfitting particularly in resource limited environments.

Research Gap

Limitations of traditional attack methodologies

Classical power-based side-channel attacks (SCAs), such as Differential Power Analysis (DPA) and Correlation Power Analysis (CPA), have been the norm in cryptographic research for decades. These techniques are, however, fraught with inherent deficiencies that make them unrealistic to apply in practice. To begin with, they demand thousands of power traces to achieve successful cryptographic key recovery, especially in side-channel attacks on unsecured DES implementations, where noise and exact sample selection make data requirements even more stringent. Second, their efficiency is extremely vulnerable to environmental noise and measurement errors since even small perturbations can complicate trace processing and, in some instances, make the attack entirely infeasible. Third, the effectiveness of these methods is highly implementation-dependent, i.e., their feasibility relies on the hardware implementation because of effects such as clock jitter, power supply variations, and algorithmic optimizations, which obliges attackers to create customized attack strategies for each target device. These limitations point to the necessity of having more efficient and flexible attack techniques-capable of working with fewer traces but being resilient to noise and implementation differences-to push the research in side-channel analysis forward.

Emergence of machine learning-based attacks

The application of machine learning (ML) methods to side-channel analysis has transformed the domain in making more effective, versatile, and adaptive attacks feasible than conventional techniques. Among the most important benefits is the prospect for launching "blind" attacks, wherein the ML models can effectively extract cryptographic keys without fine-grained details of the device implementation or ciphertexts access. For example, recent publications have shown complete key recovery on post-quantum schemes such as Kyber KEM through pure ML-based side-channel analysis, illustrating the capabilities of such methods even on contemporary cryptographic algorithms. [11] Furthermore, ML-based attacks also have the benefit of enhanced flexibility, in that they can learn complicated power consumption patterns which could potentially be overlooked by standard statistical methods, and thus prove to be effective even in noisy environments or against countermeasure-enabled devices. A second important enhancement is the decrease in trace demands, some ML techniques being able to accomplish key recovery with noticeably fewer traces compared to conventional Differential Power Analysis (DPA) or Correlation Power Analysis (CPA). Nonetheless, there are still predicaments in realizing the broad applicability of such attacks to a vast group of cryptographic implementations and enhancing the interpretability of ML models with the aim of increasing their efficiency. In spite of these open questions, ML-based side-channel analysis is a paradigm shift in the direction of a more flexible and scalable toolset for analyzing and exploiting cryptographic system vulnerabilities.

Vulnerabilities in post-quantum cryptographic algorithms

The move to post-quantum cryptography (PQC) presents novel side-channel security issues as new algorithms need to be secure not only against classical but also quantum attacks. Recent works have shown that even standardized PQC candidates such as Dilithium are susceptible to power-based side-channel attacks (SCAs) with key recovery shown on FPGA implementations through an analysis of power consumption profiles during cryptographic operations. [12] These results illustrate the algorithmic vulnerability of some PQC proposals to classical physical attacks, even though they are mathematically secure against quantum computing. In addition effective side-channel security in PQC is very difficult to realize owing to sophisticated mathematical structures and operational characteristics security, performance and resource efficiency need to be wellbalanced which is still an open problem. These advancements underscore the dire necessity for comprehensive side-channel assessments of PQC algorithms, as well as for standardized implementation guidelines offering physical security without any compromise in functionality. As the cryptographic community moves forward with quantum-resistant standards, high side-channel resistance needs to be given serious pr

Integration of countermeasures and their effectiveness

Coming up with effective countermeasures against power-based side-channel attacks (SCAs) continues to be a pressing issue, especially with the increasing sophistication of attack techniques. Although conventional defenses such as masking and hiding have been widely implemented, their effectiveness is diminishing against the more recent approaches—specifically, machine learning (ML)-based attacks, which in various research works have succeeded in breaking through these defenses. This susceptibility calls for the development of more secure defense mechanisms capable of resisting contemporary analysis methods.

Furthermore, deployment of countermeasures typically comes with considerable computational and memory overhead, posing implementation challenges for resource-limited IoT devices where security must be traded off against performance. Added to these challenges is the absence of standard testing methodologies, with researchers and practitioners lacking definite benchmarks to measure the efficacy of countermeasures against conventional and ML-based SCAs. To bridge these gaps, the cryptographic community needs to engage in developing novel, lightweight countermeasures with high security without imposing unnecessary overhead on resources, so that they are suitable for the next generation of constrained and high-performance devices.

Real-world applicability and practical constraints

Bridging the theoretical side-channel analysis-to-practice divide is challenging due to the abundance of practical considerations. Environmental variability e.g., electrical noise, temperature variation, device aging can influence both countermeasure performance and attack success unpredictably in non-laboratory environments, making it hard to reproduce. In addition, the sheer diversity of devices in operation today, with their diverse hardware architectures and software realizations, renders it challenging to develop universally effective attack methods or countermeasures. Real-world operational factors introduce a further level of challenge, in that attackers might have only intermittent physical access to targets, narrow time windows, or incomplete knowledge of system internals circumstances not usually accounted for in academic work.

These difficulties highlight the sheer importance of testing side-channel vulnerabilities and countermeasures on realistic heterogeneous setups so that their usability is guaranteed beyond idealized lab environments. In the absence of such real-world testing, even theoretically viable attacks and countermeasures can fail when realized on actual systems in the field.

Ethical considerations and responsible disclosure

As side-channel attack techniques like power analysis against cryptos like DES, become more sophisticated and simpler to apply, there is an increasing need to address the ethical considerations of their research, utilization, and disclosure. While this type of research is necessary to discover vulnerabilities in existing cryptographic systems and improve overall security it can also be exploited by malicious users if not addressed responsibly.

One of the biggest concerns is striking a balance between openness and security. On one hand, providing full methodologies and results guarantees scientific integrity and peer validation. On the other, publicly disclosing exact attack methods, especially those that can compromise widely deployed legacy systems like DES (which is still present in some embedded and legacy installations), could potentially expose real-world systems to attacks if no adequate mitigation channel is provided.

There is also increasing need for formal responsible disclosure procedures. Researchers who discover vulnerabilities through side-channel attacks ought to adhere to a standard reporting convention typically arranged with vendors, cryptographic library implementers and security standards bodies (e.g., NIST or ISO/IEC) in order to provide them time to patch or deprecate vulnerable infrastructure. [13] Sadly, most existing side-channel attack papers don't discuss such activities in any consistent way, so there are no established best practices for ethical behavior in this research community.

A second difficulty is the dual-use nature of side-channel tools. The platform developed in this paper a platform to record and classify power traces with machine learning is equally valuable for educational and for nefarious purposes. This dual-use problem demands the creation of use policies, ethics training modules, and potentially even license agreements for sophisticated attack tools used outside the academic context.

Research Problem

As cryptographic algorithms in use today keep getting stronger and more sophisticated so do the techniques for breaking them. Whereas algorithms such as DES (Data Encryption Standard) have mathematical weaknesses well documented, the most treacherous and least suspected threats do not stem from algorithmic logic weaknesses but rather from physical implementations of the algorithm in hardware. Among the many exploitation methods is power-based side-channel analysis (SCA) a highly effective method that leverages differences in the power usage of an instrument when it performs cryptographic processing for leaking sensitive data like secret keys or plaintext information.

Side-channel attacks circumvent classical cryptographic assumptions by taking advantage of realistic leakage sources. For DES, every iteration of the algorithm generates new computational patterns that are measurable as observable power traces when the algorithm is executed on microcontrollers or hardware accelerators. The traces, if captured and examined, can establish vital correlations to the internal data being processed including the intermediate states and secret key ultimately. This extraction technique has been shown to be feasible even for legacy algorithms when implemented in susceptible hardware setups. [14]

Even though power-based SCAs are of great academic concern and increasing practical significance, their actual deployment and experimentation remain relatively unexplored, especially in the context of integration with machine learning (ML) algorithms and real-time embedded systems. Most of the available literature on SCAs is limited to either simulation or high-end laboratory setups using oscilloscopes, FPGAs, or software-level trace injection utilities. These setups, though enlightening, don't capture the constraints and virtues of low-cost, ubiquitous boards like the Raspberry Pi or ESP32, which might have the potential to bring side-channel experimentation and learning within reach.

Furthermore although machine learning classifiers like multilayer perceptrons (MLPs), convolutional neural networks (CNNs) and other deep learning architectures have demonstrated potential in side-channel classification tasks the models are seldom tested or deployed on actual data captured from hardware in noisy, resource limited settings. This constitutes a research gap between theoretical side channel model performance and real time vulnerability detection in embedded systems.

Also traditional countermeasures such as masking, shuffling, and noise injection do not typically thwart ML based attacks because the models can learn complex, high-dimensional patterns and be resilient to injected randomness. As embedded systems become ubiquitous in consumer electronics, automobiles, IoT devices, and military system the need for real-time, low cost, and lightweight SCA analysis tools is greater than ever not only for performing attack research but also for enhancing the design of secure hardware.

Research Objective

Major objective

The major goal of this project is to develop, implement, and verify a low-cost, fully embedded system capable of conducting power based side channel analysis (SCA) of cryptographic operations on the Data Encryption Standard (DES) with real time data acquisition, classification of side-channel traces via machine learning, and handheld execution to verify hardware vulnerability. This is expected to close the theoretical cryptanalysis attack models and their actual, hardware-level exploitation in embedded systems.

Power based side channel attacks work by monitoring the power consumption of cryptographic devices while they are in the process of encryption or decryption. Power traces can therefore potentially leak information regarding the plaintext or secret keys being processed. While such attacks have been demonstrated to be feasible under controlled laboratory conditions using high end hardware there is a tremendous need to study how such methods can be achieved on low power off-the-shelf platforms such as Arduino, ESP32, and Raspberry Pi using sensors such as INA219 to make real-time measurements.

Sub objectives

- To design a hardware system that can measure real-time power, voltage, current, and energy consumption while encrypting with DES encryption with the help of INA219 sensors and microcontrollers like Arduino Nano and ESP32. This includes circuit designing, synchronizing with cryptographic execution, and precise data logging.
- To develop a Bluetooth-controlled logging interface based on the HC-05 module and mobile applications like Serial Bluetooth Terminal, to remotely start (STARTLOG), stop (STOPLOG), and retrieve (GETLOG) side-channel data logging without physical hardware access.
- For preprocessing and extracting statistical features (mean, max, min, standard deviation) from power traces gathered to prepare a clean dataset for machine learning models' training.
- To train and model machine learning models, in the form of Multilayer Perceptrons (MLPs), to be used for classifying and predicting sensitive cryptographic plaintext and key, and verify the accuracy of such models in modeled and actual embedded systems (Lerman et al., 2014).

- To deploy the model developed on a Raspberry Pi embedded system, a complete platform with the capability of live inference and output on an onboard 16x2 LCD display. To evaluate the robustness, precision, and usability of the overall system, creating operational challenges in noisy environments and suggesting improvements for practical, large-scale side-channel assessment tools.

METHODOLOGY

Hardware Design And Implementation

Part 1

As in the initial phase of the hardware development focus on the design the system which can capable of capturing power consumption variations and to identify the power fluctuations which happened during encryption process can be captured during cryptographic operation which is specifically to analyse the power base side channel attacks. And this hardware implementation provide insights into the vulnerabilities of the cryptographic algorithms when they executed embedded hardware. And who achieved this goal and how do you know microcontroller was used to execute the data encryption standard or the DES algorithm while the power traces the record using shunt resistor and oscilloscope. The primary objective of this hardware implementation is to identify how the encryption process influence or effect put the power usage in real time.

Components used

- Arduino Uno: A microcontroller board based on the ATmega328P, used to run DES encryption and facilitate data collection.
- Shunt Resistor: A low-value resistor placed in series with the power supply to measure current variations.
- Oscilloscope: Used to capture and visualize power traces during cryptographic operations.
- Power Supply: A regulated power source providing stable voltage to the Arduino Uno and its connected peripherals.

The hardware setup have several components which I have mentioned about for capturing the power traces. Naa do you know Uno is widely used microcontroller based on the ATmega328P was selected for the easy of implementation. The shantaram sister has use to measure small voltage variation resulting from changes in current consumption. This is an essential aspect of side channel attack analysis because there are only small fluctuations which can observe during this encryption processes. When to get visually output and records and to see the fluctuations of the power in real time we have used an oscilloscope. And also regulated power supply provided to take a stable voltage to the audio no no to prevent external oxygen from affecting the measurements and this components provide a basic set up for conducting initial experiments and to understand the power fluctuations which are going on during power base section attacks analysis. Below is the circuit diagram of this implementation.

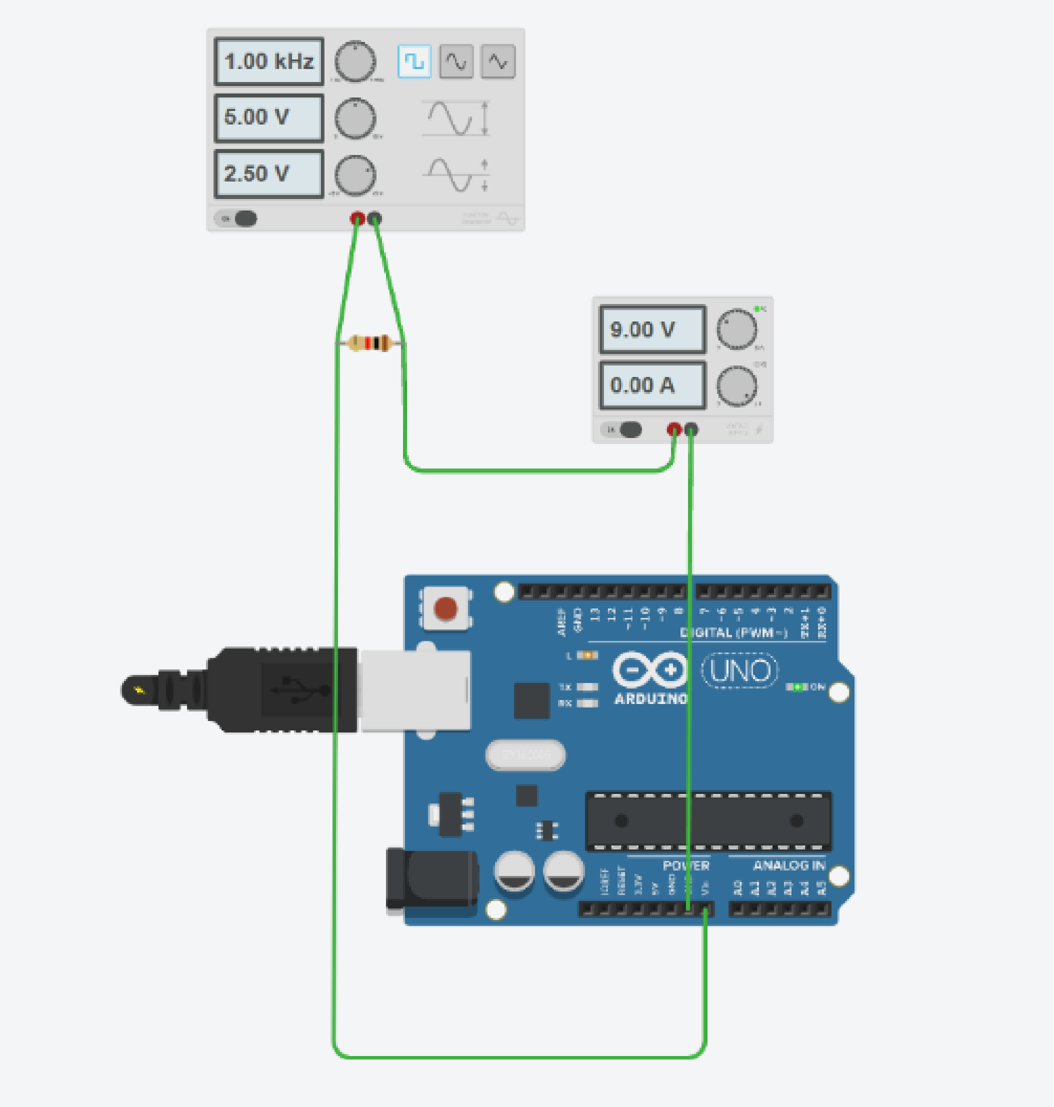


Figure 1Hardware Setup for Capture Power Traces (Arduino Uno)

Experimental setup

To capture the power variation the shunt resistor which was placed in series with the power supply line of the Arduino Uno and with the help of oscilloscope it helps to measure the voltage drop across it. The method which used is widely utilized in power analysis research to infer the current indirectly. The encryption process which is implemented on the Arduino Uno where are we given plain text and the secret key and it given an output of ciphertext.

Observations

So the recorded power traces which we saw on oscilloscope shows the encryption off the days and these variations confirm the feasibility of the power analysis attacks. And it demonstrate miserable relationship between cryptographic operation and power consumption. While the Arduino Uno provided a useful starting point, its limited sampling rate and computational power indicated the need for a more advanced platform in future iterations.

Part 2

So in here to effective analyse the power besides an attack I have created a dedicated power monitoring setup which was built using Arduino. Previously I have used Arduino Uno and when it comes to this hardware monitoring setup I have developed it into Arduino nano microcontroller. So the objective of this system east to monitor the power consumption characteristics offer separate encryption running device and to extract meaningful side channel data which are correlated for the cryptographic operations. So the as a separate device or the targeted device I have used Arduino Uno. The hardware components and the configurations are carefully selected for this to get a higher precision or higher accuracy real time responsiveness.

The main component of the heart of this implementation is depend on the INA 219 current sensor module which was designed to take the readings off high side voltage and current monitor. This is become the crucial component because it provide direct digital readings such as bus voltage shunt voltage current power through the I2C protocol. The INA219 can measure voltages up to 26V and currents up to 3.2A thanks to its integrated 12-bit analog-to-digital converter (ADC). It is set up to track the voltage drop across a 0.1-ohm shunt resistor in this configuration.

At the initial step the power monitoring system display the real time power voltage and energy fluctuations in the 0.96" I2C OLED display which give better understanding to the user about the encryption process as a visual feedback. Even though this setup was limited to a particular device observation and lacked remote access capabilities. And to identify the patterns and to get a better idea of how the fluctuations are going on this oled display is not enough it is just forget a user an idea of whether monitoring component is working. To overcome this limitation and enable the remote interaction and monitoring oled displays removed and replaced with HC 05 bluetooth module. And also there is a reason because when we at the overly display module the Arduino code will be particularly bigger and it will become an issue when storing the code in Arduino nano with the limitations of the storage. And with this modification it allowed Bluetooth or the wireless communication between or do you know and a smartphone or pc using serial Bluetooth terminal application The C05 Bluetooth module was configured with the following wiring;

RXD → D3

TXD → D2

GND

VCC → supply

The software on the Arduino side utilize the software serial library to manage the Bluetooth Communication alongside primary I2C operation. And also for the INA219 current sensor Adafruit_INA219.h was used. So to work INA219 dc current sensor The

Adafruit_INA219.h Library was used. All the functions and the constants which are required to facilitate the interaction with the sensor particularly for the Arduino this library is need. With the help of this library it makes the use of current sensor more simple and effective way when it comes to experimenting or tracking power usage.

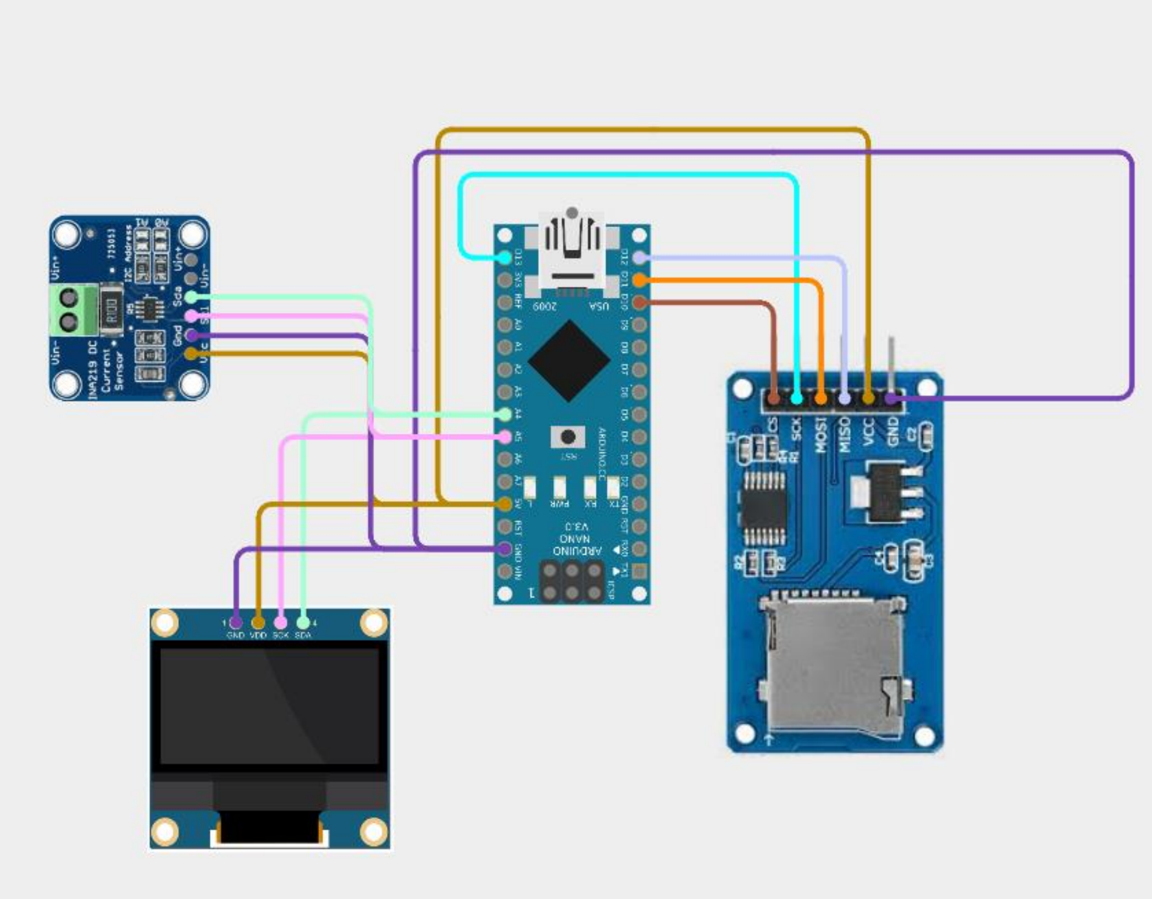


Figure 2 Initial Hardware Setup with OLED Display and SD Card

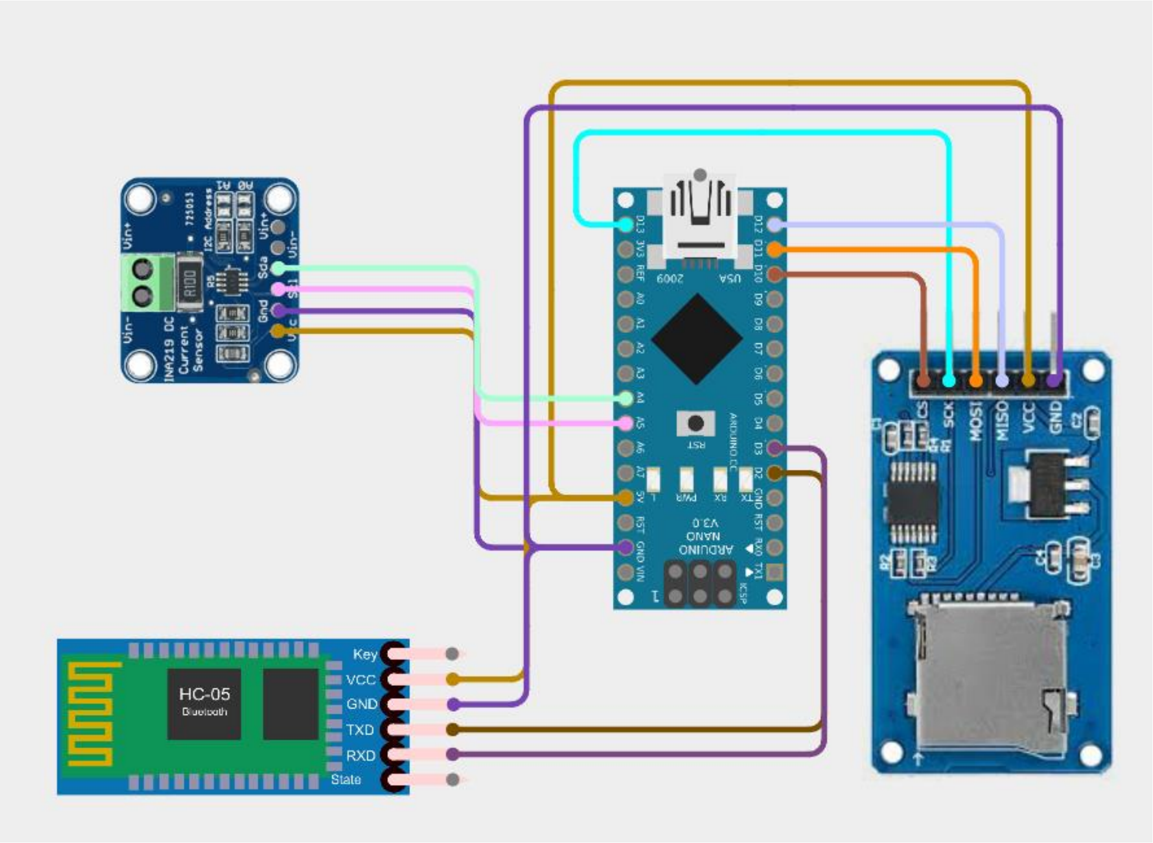


Figure 3 Improved Setup with HC-05 Bluetooth Module

Measurement principles

Using the current sensor it measures several key electrical parameters of the encryption process as below,

- **Bus Voltage (V)**: The voltage supplied to the encryption device.
- **Shunt Voltage (mV)**: The voltage drop across a known shunt resistor (0.1 ohms).
- **Current (mA)**: Derived using Ohm's Law: I=VshuntRshuntI = \frac{V_{shunt}}{R_{shunt}}
- **Power (mW)**: Calculated by multiplying the bus voltage and the measured current: P=Vbus×IP = V_{bus} \times I
- **Energy (mWh)**: Computed over time using: E=∑(P×Δt3600)E = \sum \left(\frac{P \times \Delta t}{3600} \right)

These parameters are calculated for 1 second intervals which ensuring the adequate resolution to capture the dynamic changes in power consumption during encryption. If the improvement of this to 100 millisecond of intervals we can get more accurate values but when we get the output to the serial Bluetooth terminal there is a small issue occurred a because the speed and amount can data feed via Bluetooth is limited. The calculator energy gives a cumulative view of power usage throughout the process. In the Arduino Court using the following functions from the Adafruit_INA219.h Library used to calculate these power current voltage and energy.

```
void readPowerData() {
  shuntVoltage = ina219.getShuntVoltage_mV();
  busVoltage = ina219.getBusVoltage_V();
  current_mA = ina219.getCurrent_mA();
  power_mW = ina219.getPower_mW();
  loadVoltage = busVoltage + (shuntVoltage / 1000);

  if(logging) {
    energy_mWh += (power_mW * interval) / 3600;
  }
}
```

So as you can see this code snippet shows how the voltage current and the power are sampled and how the energies cumulatively calculate using the time difference between the iterations and this measure measurements which are gathered become the basis of the side channel data which are used to the further analysis and identify plain text or the key which were used in particular encryption process.

Data logging and communication

With the enhancement of the Bluetooth module the system introduced a command based interface through serial Bluetooth terminal app and the user can give particular commands get the output. So when the user give the command **STARTLOG** via Bluetooth the Arduino system will initiate data logging by creating O overwriting data to the csv file on the SD card. According to the Arduino Code the file which created is datalog.csv. And this file include the errors of voltage current power energy and the time stamp and the sensor data recorded in one second intervals. The user and use command when the encryption process is about to start and after to stop this by giving the commands **STOPLOG** the system terminate the login process. And finally user can give the command **GETLOG**, Then this command is triggered the data written to the CSV file will be show entirely in the log file back to the terminal which give user to inspect the record data remotely.

The acquired data was written to SD card using the MicroSD card module which was implemented in the Arduino uno and this data acquisition mechanism interfaced via SPI Protocol. With the help of this SD card module it provide a persistent storage a large volume of time series sensor data. Because of the limitations of audio and Nano which are memory and processing power It was directly written to the SD card in each loop iteration to prevent the memory overflow. This Arduino system is responsible for the task that was carefully structured to read sensor values, compute derived metrics, and store them with accurate timestamps, ensuring precise alignment with the encryption activity being monitored.

Software implementation

When it comes to the software development side the Arduino was Programmed using Arduino IDE By using several libraries integrated including wire.h for I2C communication a with INA219. And use Adafruit_INA219.h for simplified Sensor access and SD.h for sd card file management and to manage data write and rewrite functions. For the Bluetooth interaction SoftwareSerial.h is used. And the main loop is handled come on passing data collection energy computation finally writing and data transmission seamlessly. Due to this this system of the power measuring hardware more robust and responsive.

The significant feature which was improved in this version of the system is shift from local overly display based live feedback to remote Bluetooth connection and get live feedback. Due to this improvement it change dramatically the particularly and the flexibility of the setup and it will allow the other researchers and the university students to monitor an experiment the real time without being physically near the device. And also additionally this modification facility the integration of monitoring system with further analytical tools for the future with using pco a mobile device and it enabling richer post processing of the side channel data.

And this entire setup was calibrated and tested by observing power profile of a known encryption routine which was executed in a separate Arduino uno which is like a white box perspective testing. Then after the capture data which consisting of voltage current power and the calculated energy exhibited the distinct fluctuation that align with the computational faces of the encryption algorithm. These variations are take a critical part because they potentially reveal the patterns that can be exploited inside channel attacks. And for the university students and researchers they can identify how those fluctuations are related to the encryption process.

Finally the conclusion this design system is provided effective adaptable means of collecting power consumption data during encryption process and by integrating necessary and precise current sensors, Robust datalogging and wireless communication system is well suited for site channel attack experiments. The methodology which used to get the real time Bluetooth live feedback is one step a put forward shift from manual sd card analysis. And this ensure that the system met the demands of the practical flexible and effective research purposes for the researchers and the university students. And this will can be implemented for other types of algorithms to analyze their behavioral patterns and get a better idea.

Why ESP32

Why ESP32 is better than Arduino nano is There are some several features in some advancement of cpu clock speed and memory. Below are some key features of ESP32 when it compares to Arduino nano.

Feature	**ESP32**	**Arduino Nano**
CPU	Dual-core 32-bit Xtensa LX6	Single-core 8-bit ATmega328P
Clock Speed	Up to 240 MHz	16 MHz
Wi-Fi	Yes (built-in)	No
Bluetooth	Yes (Bluetooth + BLE)	No
Flash memory	4MB (varies by board)	32KB (of which ~2KB used by bootloader)
SRAM	520KB	2KB
GPIO Pins	~30 (depends on model)	14 digital, 8 analog
ADC Resolution	12-bit	10-bit
PWM	Yes (more channels)	Yes
DAC	Yes (2 channels)	No
Touch Pins	Yes	No
Power consumption	More powerful, but higher power usage. Has deep sleep modes for battery efficiency in IoT.	Very low power by default, ideal for ultra-low power projects if features are limited.

Table 1 ESP32 vs Arduino Nano

Part 3 – methodology (hardware enhancement with ESP32)

So as the third stage of the hardware development process of the power based side channel attack analysis system was transitioning from Arduino Nano to ESP32. This decision was taken because of some limitations of the Arduino Nano which faced before. Those limitations are like memory capacity computational power and lack of built in wireless communication capabilities, Those limitations or the difference between the ESP 32 and the Arduino nano have mentioned previously. The ESP32 low cost the low power system on a cheap with the integrated virus connection which are Wi-fi and Bluetooth offering a powerful and scalable solution to handle this data acquisition and communicating. Below are the motivations to upgrade the system from Arduino nano to ESP32.

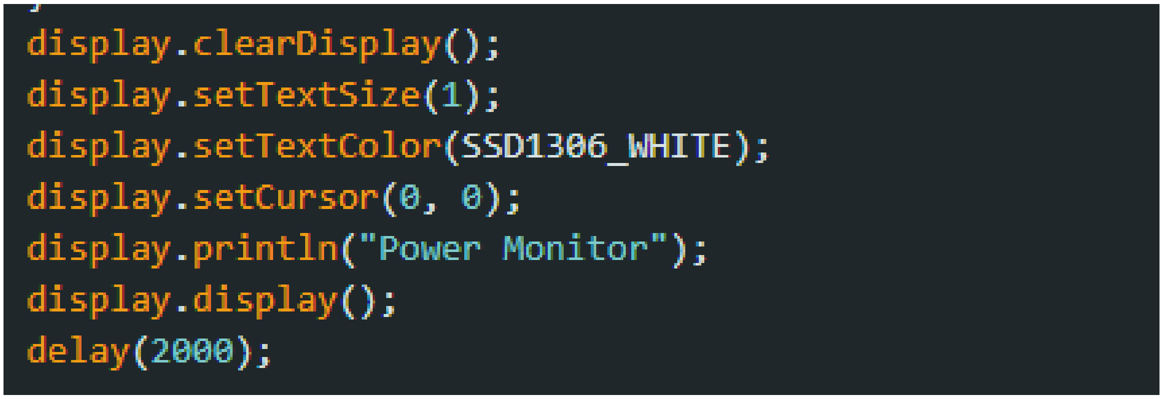
Motivation for upgrading to ESP32

Memory constrains

When it comes to the memory constraints the Arduino Nano which have only 32KB Flash memory and 2KB of SRAM which was mentioned previously. The two kilobyte SRAM is insufficient to store and execute the expanded data login and display features required for real time monitoring. And with that storage limitation either OLED display module or Bluetooth terminal can be replaced. Because OLED module helps to identify whether the device is functional correctly. As in contrast the ESP32 features up to 4 MB of flash memory and 520 KB of SRAM. This will improve the capacity to handle multiple libraries like Wire.h, Adafruit_INA219.h, Adafruit_SSD1306.h, and SD.h. and to store more code. And this memory expansion is very critical because accommodating larger buffer sizes storing sensor readings managing display operation and handling Bluetooth serial commands.

Real time display with OLED

As mentioned previously the OLED module was integrated into the system to provide real time visualization of power current voltage and energy consumption data and it helped for the users to identify whether device is correctly working. This OLED module have 128x64 pixels. And this is connected using I2C interface which is SDA or SCL. This allowed the users to monitor the encryption process behavior and it enabling easier debugging and more interactive analysis. In the display updates are shown using the functions display.print() and display.display() from the Adafruit SSD1306 library.



```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(0, 0);
display.println("Power Monitor");
display.display();
delay(2000);
```

Figure 4 OLED Module Arduino configuration

Wire less communication

With the upgrade of ESP32 it eliminates the need of external Bluetooth modules like HC 0 5 using the Library of BluetoothSerial.h. This library allow serial Bluetooth interface enable commands such as STARTLOG, STOPLOG, and GETLOG to be issued via serial Bluetooth terminal mobile app. This feature allows the remote control capability of controlling the logging process and transmission of data which are recorded over Bluetooth in plain text format.

Increased processing power

There is P32 is having a dual core Xtensa LX6 processors running at up to 240 MHz which is compared to the Arduino Nano which have a 16 MHz ATmega328P. With this considerable speed difference or the processing speed allows more faster I2C data polling, real-time OLED display refresh, and efficient SD card I/O operations. As side-channel attacks require time-sensitive data sampling, the real-time performance of ESP32 was crucial.

Updated circuit diagram

The upgraded hardware now have following key components,

- ESP32 Development Board
- INA219 Current Sensor
- OLED I²C Display (128x64)
- SD Card Module
- Load (Encryption running Arduino Uno)

Below are the wiring configurations of the system,

INA219 to ESP32	SDA → GPIO 21 SCL → GPIO 22 VCC → 3.3V GND → GND
OLED Display	SDA → GPIO 21 SCL → GPIO 22 VCC → 3.3V GND → GND
SD Card Module	MISO → GPIO 19 MOSI → GPIO 23 SCK → GPIO 18 CS → GPIO 5 VCC → 3.3V GND → GND

Table 2 ESP32 connections

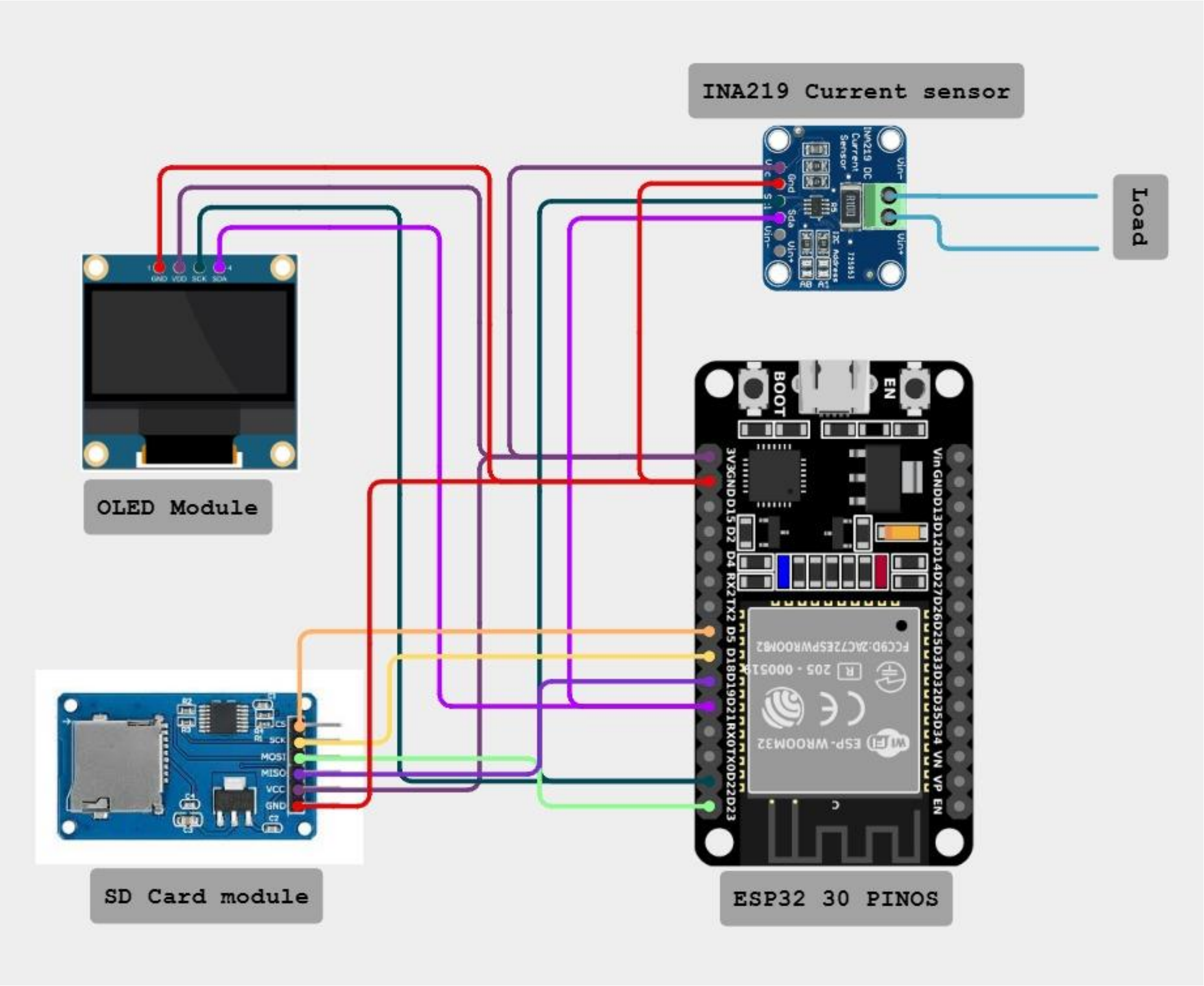
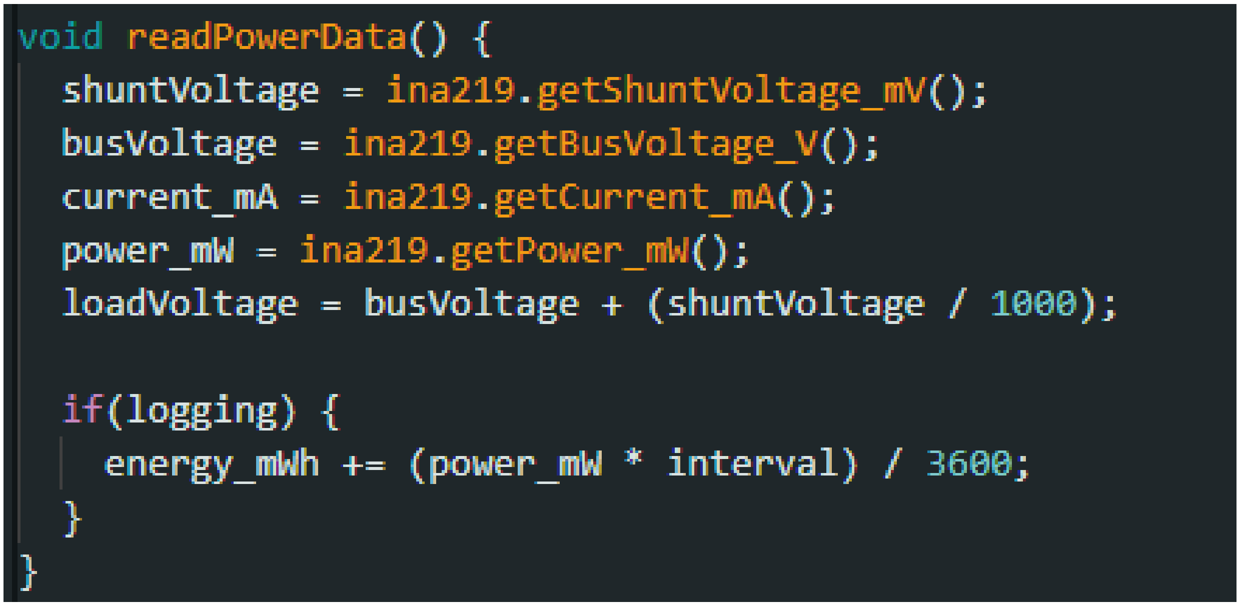


Figure 5 Improved Setup with ESP32

Key software functions

Real-time power measurement and energy accumulation

This function is the core mechanism which used to measure power consumption during encryption process and the system uses a current sensor which provide high side current voltage and power monitoring. And this function collecting real time electrical parameters which are both voltage across shunt register and bus voltage to derive the power consumed by encryption device below the detail breakdown of how the functions are working to calculate and obtain.



```
void readPowerData() {
  shuntVoltage = ina219.getShuntVoltage_mV();
  busVoltage = ina219.getBusVoltage_V();
  current_mA = ina219.getCurrent_mA();
  power_mW = ina219.getPower_mW();
  loadVoltage = busVoltage + (shuntVoltage / 1000);

  if(logging) {
    energy_mWh += (power_mW * interval) / 3600;
  }
}
```

Figure 6 Energy accumulation function

1. shuntVoltage = ina219.getShuntVoltage_mV();
    - Measures the voltage drop across the shunt resistor in millivolts. This drop is directly proportional to the current flowing through the device.

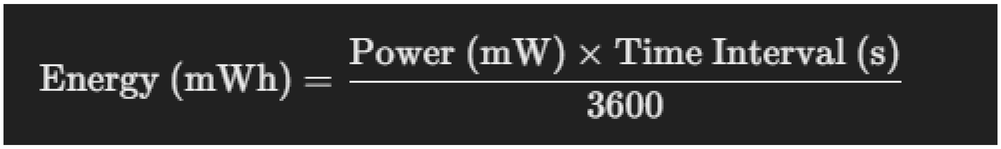
2. busVoltage = ina219.getBusVoltage_V();
    - Measures the voltage from the INA219's input pin to ground, which represents the system voltage being supplied to the encryption device.

3. current_mA = ina219.getCurrent_mA();
    - Retrieves the actual current drawn by the encryption module, calculated internally by the INA219 using the shunt resistor value and the shunt voltage.

4. power_mW = ina219.getPower_mW();
    - Retrieves the real-time power consumption in milliwatts. This value is derived as voltage × current.

5. loadVoltage = busVoltage + (shuntVoltage / 1000);
    - Adds the voltage drop across the shunt resistor to the bus voltage to get the total voltage experienced by the load.

6. Energy Calculation Logic:
   - When logging is enabled (if(logging)), energy is accumulated by integrating power over time using the formula:

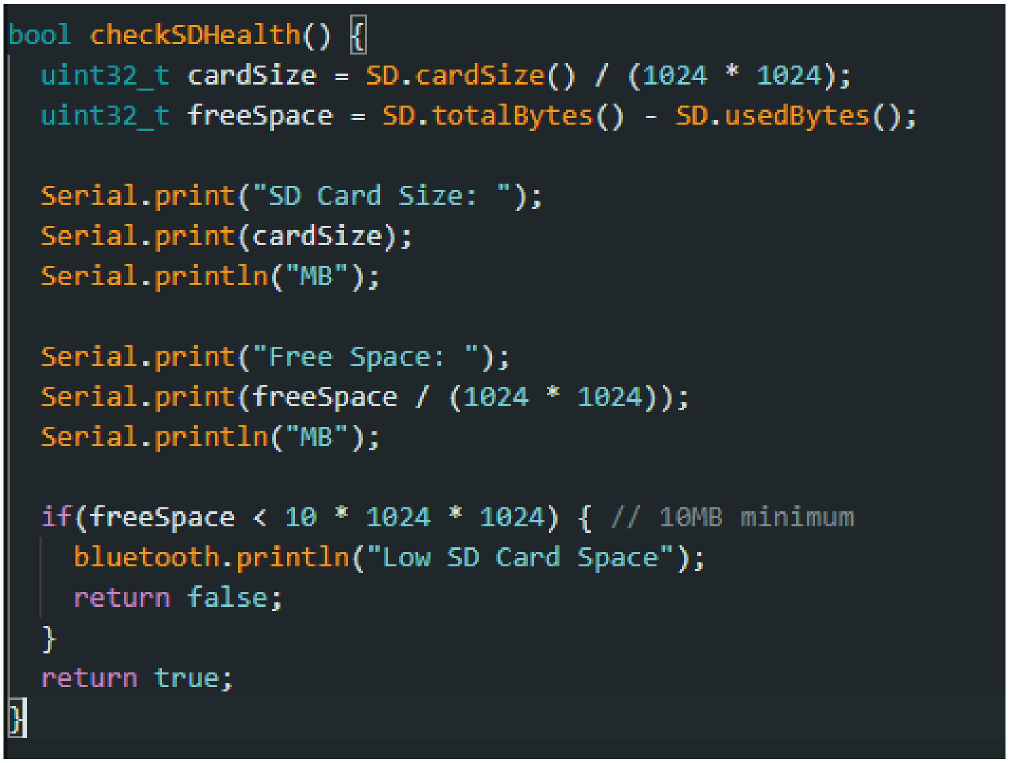


$$\text{Energy (mWh)} = \frac{\text{Power (mW)} \times \text{Time Interval (s)}}{3600}$$

Figure 7Energy Calculating Equation

Monitoring storage capacity for logging

So when data storing to the sd card there is a small issue faced which is When there is insufficient space there are some file reading issue and also due to create some garbage memory this incident was happened So as a mitigation procedure or a proactive health check methodology is applied to this through this function. So the basic function of this is check if there are available space for data logging before proceeding with writing operation below are the function breakdown



```
bool checkSDHealth() {
  uint32_t cardSize = SD.cardSize() / (1024 * 1024);
  uint32_t freeSpace = SD.totalBytes() - SD.usedBytes();

  Serial.print("SD Card Size: ");
  Serial.print(cardSize);
  Serial.println("MB");

  Serial.print("Free Space: ");
  Serial.print(freeSpace / (1024 * 1024));
  Serial.println("MB");

  if(freeSpace < 10 * 1024 * 1024) { // 10MB minimum
    bluetooth.println("Low SD Card Space");
    return false;
  }
  return true;
}
```

Figure 8 SD card Health checking function

1. SD.cardSize() and SD.totalBytes() - SD.usedBytes()
   - These functions compute the total capacity of the SD card and the currently available free space. The result is converted to megabytes (MB) for easier readability and system management.

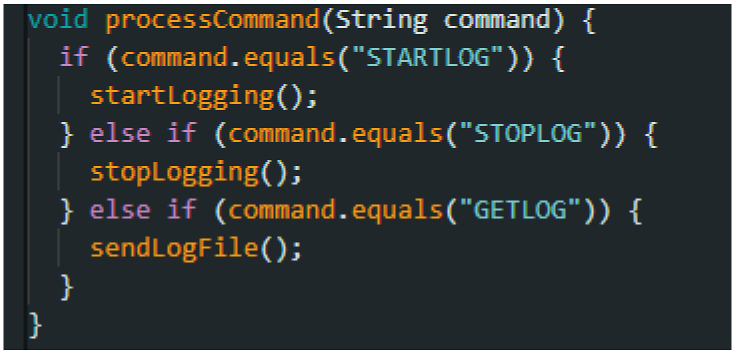
2. Serial Output
   - The function prints both the total SD card size and the remaining free space to the serial monitor. This is useful during debugging and testing phases.

3. Bluetooth Warning
   - If the free space falls below 10 MB, a warning is sent to the connected Bluetooth terminal (bluetooth.println("Low SD Card Space")). This allows the user to take action (e.g., replace or format the SD card) before a failure occurs.

4. Return Value
   - The function returns false if the space is insufficient and true if it is healthy, enabling the main program to conditionally start logging only if the SD card is in good condition.

Bluetooth command handling and logging control

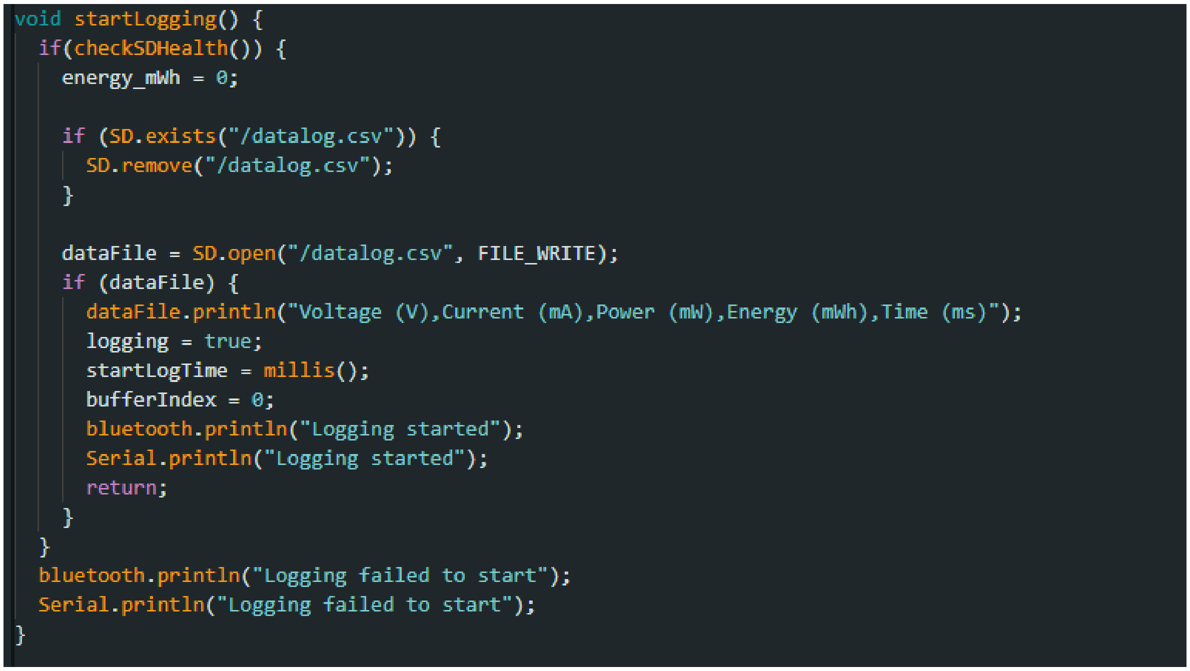
And one of the critical enhancements in the ESP32 based system which first talked before is addition of Bluetooth Command interface which is providing remote management of data logging with the help of simple commands which were given in the code as STARTLOG, STOPLOG and GETLOG and this commands are identified and processed by a master function processCommand(). And this function takes a command in string format which was given by the user and verify it against predefined options and forward to the corresponding operation. So this modular design make it so easy to add or change the command in the future releases of firmware enabling the system to grow and to be adapt. And the wireless communication which was implemented in the system helps to increase the usability and the accessibility especially in lab situations where it is tricky to handle the device.



```
void processCommand(String command) {
  if (command.equals("STARTLOG")) {
    startLogging();
  } else if (command.equals("STOPLOG")) {
    stopLogging();
  } else if (command.equals("GETLOG")) {
    sendLogFile();
  }
}
```

Figure 9 Process String Command function

So in the function start logging initiate logging in a safe manner which upon receiving the STARTLOG command this function does a quick health check on the street card using the checkSDHealth() function Which was described previously that is used to check whether there are sufficient space is available in the sd card. Otherwise it's not checked there may be failure in the system due to low memory. And if there is a present log file it will be removed and then start to write to avoid problems or data corruption. The new file is greater than initiated with header row of label columns Voltage , Current , Power , Energy , and Time (ms)for convenient analysis after the experiment. Important logging variables like energy_mWh, startLogTime, and bufferIndex are reset, and the logging state is enabled. Then after the system sends live feedback to the user via the Serial Monitor and the Bluetooth link that logging has started successfaully. If there is any malfunction during initialization, error messages are sent to notify the user.



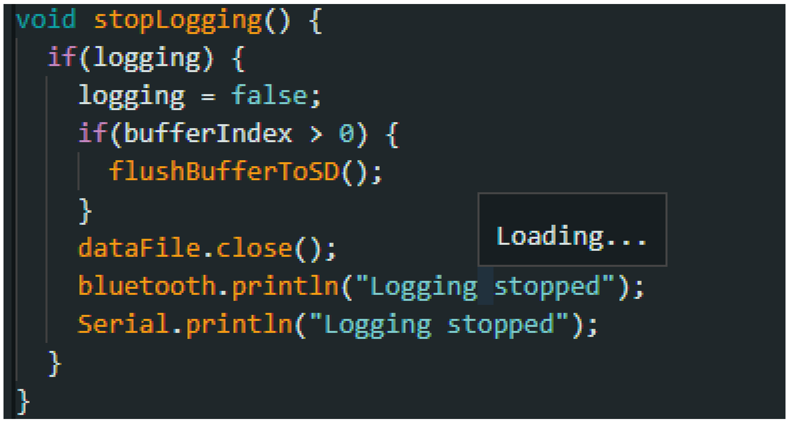
```
void startLogging() {
  if(checkSDHealth()) {
    energy_mWh = 0;

    if (SD.exists("/datalog.csv")) {
      SD.remove("/datalog.csv");
    }

    dataFile = SD.open("/datalog.csv", FILE_WRITE);
    if (dataFile) {
      dataFile.println("Voltage (V),Current (mA),Power (mW),Energy (mWh),Time (ms)");
      logging = true;
      startLogTime = millis();
      bufferIndex = 0;
      bluetooth.println("Logging started");
      Serial.println("Logging started");
      return;
    }
  }
  bluetooth.println("Logging failed to start");
  Serial.println("Logging failed to start");
}
```

Figure 10 Logging starting function

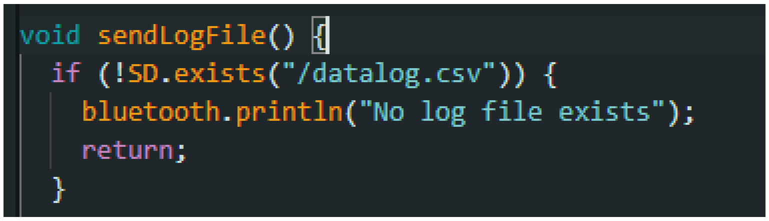
When we need to come to the stopLogging function when it called response to the "STOPLOG" command and plays a critical role in safely terminating the logging session. If logging is active, the function disables further data capture by resetting the logging flag and flushes any unsaved data from the temporary buffer to the SD card using flushBufferToSD(). And this will ensure the data integrity there are some cases the power loss of communication errors might otherwise result in data loss. After this it closes the csv file the logging session and sends confirmation messages to the user via both Bluetooth and Serial output. This controlled shutdown mechanism is crucial for protecting the collected data and maintaining the reliability of the dataset.



```
void stopLogging() {
  if(logging) {
    logging = false;
    if(bufferIndex > 0) {
      flushBufferToSD();
    }
    dataFile.close();
    bluetooth.println("Logging stopped");
    Serial.println("Logging stopped");
  }
}
```

Figure 11 Logging stop function

And finally the sendLogFile() function is triggered by the GETLOG command and serves the purpose of retrieving the logged data wirelessly And show the data written in the terminal so the user can identify how the power fluctuations are happened. Before send the file the function first of all check for the existence of data log and if there is no any informative message will send to the user. And if it is this typically read line by line and transmitted over terminal interface which I have told you before although the implementation of the function is not shown in the snippet. This capability allows the user to access and analyze data immediately without the need to physically remove the SD card, significantly streamlining the experimental workflow.



```
void sendLogFile() {
  if (!SD.exists("/datalog.csv")) {
    bluetooth.println("No log file exists");
    return;
  }
```

Figure 12 Saved file Reading function

Dataset Generation

For the data set generation Series of real time power measurements capture during the DES encryption operations. This take a crucial step in the data model training process and these measurements are collected by the customary design order setup which was mentioned previously in the methodology which have a micro controller of ESP32 and get precision current sensor INA219 monitoring sensor. And the system purpose was to build the data set To analyze power based side channel attacks by capturing the fluctuation of physical parameters which are voltage current power and energy during a particular cryptographic process.

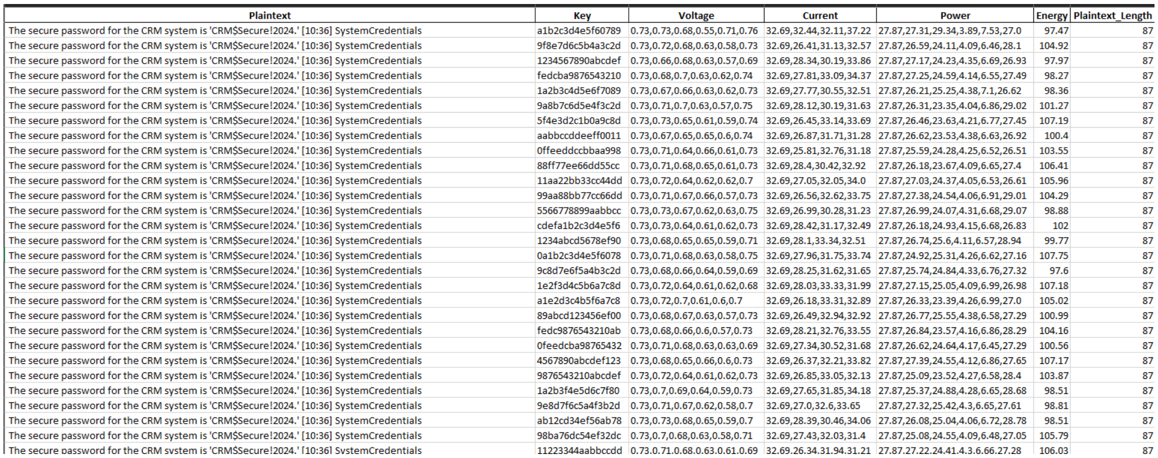
When taking the encryption values it carefully monitored and the measurements are recorded and stored in structured formats which are in excel format. In order to create a tabular Microsoft Excel data set which is use use for training and testing of machine learning models these are preprocessed and consolidated. This consists of sample data collection of 6500 each of the record shows a DES encryption occurrence. Below are the key attributes in the dataset,

- Plaintext: The original unencrypted text (string format).
- Key: The secret encryption key used in the DES process (hexadecimal string).
- Voltage: A string containing multiple comma-separated float values in volts, representing temporal voltage readings during encryption.
- Current: Similar to voltage, this field consists of multiple comma-separated current readings in milliamperes (mA).
- Power: Calculated power values in milliwatts (mW), also represented as a comma-separated string.
- Energy: A single scalar value indicating the cumulative energy consumption for the encryption process in milliwatt-hours (mWh).
- Plaintext_Length: An integer feature capturing the character count of the plaintext input.

When prepare in the data for modeling the raw string-based power, voltage, and current features were processed using statistical aggregation and each of the comma-separated vectors was converted into four numerical features: mean, maximum, minimum, and standard deviation. With the help of this transformation it allow for reduction of high dimensional signal data into fixed length feature vectors which making them suitable for use in dense feed forward neural networks.

Creation of the data set is aligned with broad objective of capturing side channel leakages during encryption to intercept sensitive cryptographic parameters a technique conceptually grounded in the field of physical cryptanalysis. Side-channel attacks such as Differential Power Analysis (DPA) and Simple Power Analysis (SPA), as initially proposed by Kocher et al. (1999), emphasize the exploitation of measurable hardware behavior to compromise cryptographic security

And this data set is notably significant and also encapsulate both the real-world noise and signal characteristics of embedded cryptographic operations Nix origin from a physical implementation rather than simulation ensures that model is train on authentic patterns that reflect practical vulnerabilities in hardware-based encryption. And simultaneously the data set provide a robust foundation for machine learning models aimed at identifying exploitable correlations between power consumption patterns and cryptographic parameters such as plaintext and keys.



Plaintext	Key	Voltage	Current	Power	Energy	Plaintext_Length
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	a1b2c3d4e5f60789	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	9f8e7d6c5b4a3c2d	0.73,0.72,0.68,0.63,0.58,0.73	32.69,26.41,31.13,32.57	27.87,26.59,24.11,4.09,6.46,28.1	104.92	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	1234567890abcdef	0.73,0.66,0.68,0.63,0.57,0.69	32.69,28.34,30.19,33.86	27.87,27.17,24.23,4.35,6.69,26.93	97.97	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	fedcba9876543210	0.73,0.68,0.7,0.63,0.62,0.74	32.69,27.81,33.09,34.37	27.87,27.25,24.59,4.14,6.55,27.49	98.27	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	1a2b3c4d5e6f7089	0.73,0.67,0.66,0.63,0.62,0.73	32.69,27.77,30.55,32.51	27.87,26.21,25.25,4.38,7.1,26.62	98.36	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	9a8b7c6d5e4f3c2d	0.73,0.71,0.7,0.63,0.57,0.75	32.69,28.12,30.19,31.63	27.87,26.31,23.35,4.04,6.86,29.02	101.27	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	5f4e3d2c1b0a9c8d	0.73,0.73,0.65,0.61,0.59,0.74	32.69,26.45,33.14,33.69	27.87,26.46,23.63,4.21,6.77,27.45	107.19	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	aabbccddeeff0011	0.73,0.67,0.65,0.65,0.6,0.74	32.69,26.87,31.71,31.28	27.87,26.62,23.53,4.38,6.63,26.92	100.4	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	0ffeeddccbbaa998	0.73,0.71,0.64,0.66,0.61,0.73	32.69,25.81,32.76,31.18	27.87,25.59,24.28,4.25,6.52,26.51	103.55	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	88ff77ee66dd55cc	0.73,0.71,0.68,0.65,0.61,0.73	32.69,28.4,30.42,32.92	27.87,26.18,23.67,4.09,6.65,27.4	106.41	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	11aa22bb33cc44dd	0.73,0.72,0.64,0.62,0.62,0.7	32.69,27.05,32.05,34.0	27.87,27.03,24.37,4.05,6.53,26.61	105.96	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	99aa88bb77cc66dd	0.73,0.71,0.67,0.66,0.57,0.73	32.69,26.56,32.62,33.75	27.87,27.38,24.54,4.06,6.91,29.01	104.29	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	5566778899aabbcc	0.73,0.73,0.67,0.62,0.63,0.75	32.69,26.99,30.28,31.23	27.87,26.99,24.07,4.31,6.68,29.07	98.88	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	cdefa1b2c3d4e5f6	0.73,0.73,0.64,0.61,0.62,0.73	32.69,28.42,31.17,32.49	27.87,26.18,24.93,4.15,6.68,26.83	102	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	1234abcd5678ef90	0.73,0.68,0.65,0.65,0.59,0.71	32.69,28.1,33.34,32.51	27.87,26.74,25.6,4.11,6.57,28.94	99.77	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	0a1b2c3d4e5f6078	0.73,0.71,0.68,0.63,0.58,0.75	32.69,27.96,31.75,33.74	27.87,24.92,25.31,4.26,6.62,27.16	107.75	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	9c8d7e6f5a4b3c2d	0.73,0.68,0.66,0.64,0.59,0.69	32.69,28.25,31.62,31.65	27.87,25.74,24.84,4.33,6.76,27.32	97.6	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	1e2f3d4c5b6a7c8d	0.73,0.72,0.64,0.61,0.62,0.68	32.69,28.03,33.33,31.99	27.87,27.15,25.05,4.09,6.99,26.98	107.18	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	a1e2d3c4b5f6a7c8	0.73,0.72,0.7,0.61,0.6,0.7	32.69,26.18,33.31,32.89	27.87,26.33,23.39,4.26,6.99,27.0	105.02	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	89abcd123456ef00	0.73,0.68,0.67,0.63,0.57,0.73	32.69,26.49,32.94,32.92	27.87,26.77,25.55,4.38,6.58,27.29	100.99	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	fedc9876543210ab	0.73,0.68,0.66,0.6,0.57,0.73	32.69,28.21,32.76,33.55	27.87,26.84,23.57,4.16,6.86,28.29	104.16	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	0feedcba98765432	0.73,0.71,0.68,0.63,0.63,0.69	32.69,27.34,30.52,31.68	27.87,26.62,24.64,4.17,6.45,27.29	100.56	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	4567890abcdef123	0.73,0.68,0.65,0.66,0.6,0.73	32.69,26.37,32.21,33.82	27.87,27.39,24.55,4.12,6.86,27.65	107.17	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	9876543210abcdef	0.73,0.72,0.64,0.61,0.62,0.73	32.69,26.85,33.05,32.13	27.87,25.09,23.52,4.27,6.58,28.4	103.87	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	1a2b3f4e5d6c7f80	0.73,0.7,0.69,0.64,0.59,0.73	32.69,27.65,31.85,34.18	27.87,25.37,24.88,4.28,6.65,28.68	98.51	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	9e8d7f6c5a4f3b2d	0.73,0.71,0.67,0.62,0.58,0.7	32.69,27.0,32.6,33.65	27.87,27.32,25.42,4.3,6.65,27.61	98.81	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	ab12cd34ef56ab78	0.73,0.73,0.68,0.65,0.59,0.7	32.69,28.39,30.46,34.06	27.87,26.08,25.04,4.06,6.72,28.78	98.51	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	98ba76dc54ef32dc	0.73,0.7,0.68,0.63,0.58,0.71	32.69,27.43,32.03,31.4	27.87,25.08,24.55,4.09,6.48,27.05	105.79	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	11223344aabbccdd	0.73,0.71,0.68,0.63,0.61,0.69	32.69,26.34,31.94,31.21	27.87,27.22,24.41,4.3,6.66,27.28	106.03	87

Figure 13 Model Training Dataset

Analyzing Component

Model used

As the predictive modeling component of this research Multilayer Perceptron (MLP)-based Feedforward Neural Network (FNN) was used. By this selection of this model architecture was guided by its proven capacity to approximate complex, non-linear mappings between input features and target output One of the key requirement in side-channel analysis where the relationship between power traces and cryptographic secrets is inherently non-linear and obfuscated by noise.

MLP is on the class of supervised learning algorithms which is onto the family of artificial neural networks (ANNs) which is consisting of multiple layers of nodes in a directed graph with having each layer fully connected to the next one. Each of node in the network applies a weighted sum of its inputs followed by a non-linear activation function. This is allowing the network to model non-trivial data patterns. During thid study MLP consist of an input layer, three hidden layers (with 128, 64, and 32 neurons respectively), and an output layer configured for classification via the softmax function. And this layered design facilitates hierarchical feature extraction, wherein deeper layers can capture increasingly abstract representations of the power signature data. Reason behind to choose MLP architecture is supported by its widespread use in classification tasks where the input data is structured and tabular in nature as is the case with the dataset in this work We just derived from statistical summaries of the voltage, current, power, and energy signals during encryption events. Other than the convolutional or recurrent architectures this is more suitable for spatial or sequential data respectively. This architecture is a vision when it comes to deal with lower-dimensional fixed-size input vectors obtained through preprocessing or feature engineering.

Moreover this MLP architecture is computationally less intensive compared to more complex deep learning models which is making it ideal for prototyping and rapid experimentation in embedded systems research. And because of that it allows for faster training cycles and simpler deployment pipelines, while still providing sufficient expressive power to learn subtle correlations between hardware power consumption and encryption parameters.

Previously done studies related to side channel attack analysis which have demonstrated the effectiveness of MLP when it comes to similar text. As an example Lerman et al. compared machine learning techniques for side-channel classification and found that neural network-based models outperformed traditional statistical approaches in terms of accuracy and adaptability to noisy data. [15] And also Benadjila et al. highlighted the MLP's suitability for deep learning-based side channel attacks which demonstrating that properly trained MLPs could rival more sophisticated architectures on well-preprocessed datasets. [16]

Why this model is used

There are several factors or the reasons behind to select mlt for this research to analyze power based side channel attack analysis. As the first reason the dataset used in this study made of statistical features extracted from time-domain electrical measurements specifically the mean, max, min, and standard deviation of voltage, current, power, and energy during DES encryption operations. These features extracted features are highly suitable restructured learning task which make a dense and fully connected architecture more appropriate than models intended for sequential or image-based data such as recurrent or convolutional networks.

When it comes to the nonlinear relationship between features and target classes MLP is particularly better for modeling. As in the side channel attack contexts relationship between cryptographic operations and power consumption is often indirect and obscured by noise and other external factors and feedforward network which having multiple layers of non-linear activations [17] can learn these hidden patterns more effective manner. Compare to other simpler models like logistic regression or decision trees the neural networks have shown the extra ordinary ability to generalize over high-dimensional feature spaces typical in side-channel contexts.

Other advantage of this mlp is scalability and flexibility when it comes to this project it evolves to include more complex inputs so the architecture can be extended with minimal restructuring additional layers can be added as the developer need. And moreover the use of ReLU activation functions which allows the network to train faster and avoid the issues such as as vanishing gradients and while the softmax output layer ensures accurate multi-class classification by returning probabilities for each class label.

And because of the adaptive learning rate strategy the Adam Optimizer which has been choose and it is more suited for the models with moderate number of parameters and having potentially noisy input features like power consumption. Adam combines the advantages of both AdaGrad and RMSProp optimizers, allowing the model to converge quickly and handle sparse gradients efficiently. [18]

Ultimately MLP will be effect for the balance between performance, interpretability and implementation complexity so this is easy to deploy and computationally efficient for the data volume used and provides clear outputs that support the evaluation of side-channel vulnerability. When it comes to this case whether the leakage is sufficient to recover plaintext or key information. Moreover this model have good track record on the similar domains which are power analysis attacks, side-channel leakage detection, and general tabular data classification tasks. [19]

Model architecture

The machine learning model which is used in memory architecture the type of feedforward artificial neural network designed to handle classification tasks involving structured, tabular data which is discussed previously. This model is configured to learn form the statistical features which was extracted from the side-channel traces collected during the DES encryption process. And these features represent numerical summaries (mean, max, min, and standard deviation) of power, voltage, current, and energy consumption during encryption, thus forming a fixed-size input vector ideal for MLP processing. The architecture is composed of four layers:

1. Input layer

   The input layer receives a vector of 16 engineered features derived from the physical measurements of each encryption instance. And these features are scaled using StandardScaler to normalize the data and facilitate efficient training. Normalization process ensures that the magnitude of input values does not skew the model's learning behavior.

2. Hidden layers

   This model include 3 hidden layers each fully connected to the next layer

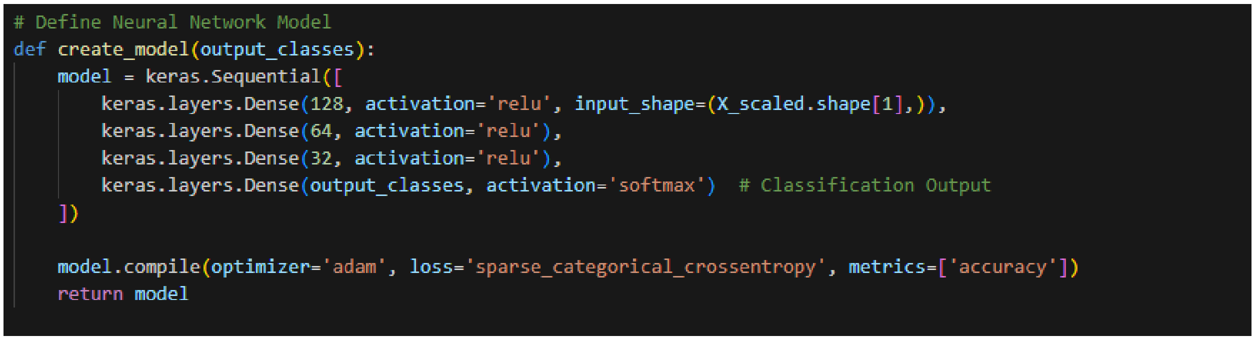
   - First Hidden Layer: 128 neurons with ReLU (Rectified Linear Unit) activation. This layer captures high-dimensional feature interactions and introduces non-linearity.
   - Second Hidden Layer: 64 neurons, also with ReLU activation refining the learned feature representation.
   - Third Hidden Layer: 32 neurons with ReLU reducing the dimensionality of the data while retaining essential features for classification.

   The ReLU activation function was chosen due to its computational efficiency and effectiveness in avoiding vanishing gradient problems during backpropagation. ReLU also promotes sparsity in the network which can help reduce overfitting and improve model interpretability.

3. Output layer

   The final layer uses a softmax activation function producing a probability distribution over the classes. For plaintext and key prediction tasks, the number of output neurons corresponds to the number of unique classes (distinct plaintext or key values) in the dataset. The class with the highest probability is selected as the model's prediction.

The architecture is defined in TensorFlow/Keras as follows,



```
# Define Neural Network Model
def create_model(output_classes):
    model = keras.Sequential([
        keras.layers.Dense(128, activation='relu', input_shape=(X_scaled.shape[1],)),
        keras.layers.Dense(64, activation='relu'),
        keras.layers.Dense(32, activation='relu'),
        keras.layers.Dense(output_classes, activation='softmax')  # Classification Output
    ])

    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model
```

Figure 14 Model Architecture Design

Decided to go with Adam for optimization seems like the right balance between adaptability and stability. It handles the learning rate adjustments automatically and combines the best parts of RMSProp and momentum-based approaches. Given how noisy power traces can be having an optimizer that adapts on the fly feels essential. Since we are dealing with integer class labels (key guesses, plaintext bytes, etc.) sparse categorical crossentropy was the obvious pick. No need for one-hot encoding, which keeps things cleaner. Tracking accuracy as the main metric simple, interpretable, and aligns with our end goal of correct key recovery.

- Epochs: 1000 – Might seem excessive but side-channel data is tricky. We need to ensure the model fully converges even if early stopping might kick in earlier.
- Batch Size: 16 – Small enough to avoid memory issues but large enough to keep gradient updates stable. Toyed with 8 and 32, but 16 gave the best trade-off between speed and consistency.
- Validation: Skipped explicit validation splits during training (for now) to keep things simple. Instead, we’ll evaluate accuracy post-training on a held-out test set. Might revisit this if overfitting becomes obvious.

The selecting of this architecture was selected is not only the simplicity and computational efficiency but also for its demonstrated ability to capture complex, non-linear relationships in power-based side-channel data. Reducing the size of the hidden layers progressively this model applies dimensional compression, which serves to extract the most salient features and reduce overfitting a common issue in side-channel machine learning tasks.

Optimization methods used and rationale

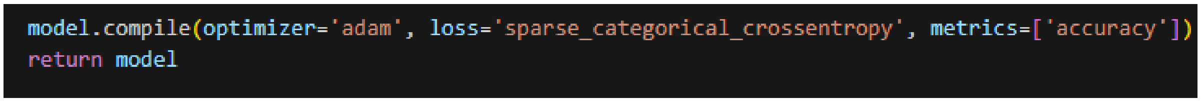
The success of a machine learning model really comes down to how you train it. For our MLP model, we used tried-and-true optimization methods to help it learn faster make more accurate predictions and work well with new data. We relied on three key ingredients: the Adam optimizer to efficiently adjust the model sparse categorical crossentropy to measure errors and standard scaling to normalize all our input features.

Adam optimizer

We chose Adam (Adaptive Moment Estimation) as our go-to optimizer for training the neural network. Think of Adam as a smart version of gradient descent that automatically adjusts the learning rate for each parameter in the model. It takes the best features from two popular methods AdaGrad ability to handle sparse data and RMSProp knack for dealing with changing patterns and combines them into one powerful package.

What makes Adam really stand out is how it customizes the learning process for every single weight in the network by tracking both the average and the variability of the gradients. This means it can find the optimal solution faster especially when working with messy complex data like the power measurements we are dealing with. Plus it's remarkably forgiving you do not need to spend ages tweaking settings to get good results which is perfect for security applications where reliability is crucial.

For our implementation we stuck with Keras default settings (learning rate of 0.001, β_1 at 0.9, β_2 at 0.999) since these have proven to work well across all sorts of different challenges. It is the machine learning equivalent of 'it just works' right out of the box.



```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
return model
```

Figure 15 Model Optimizers

Sparse categorical crossentropy loss

For training we used sparse categorical crossentropy as our loss function basically the go to choice when you're dealing with multiclass classification using simple number labels (like class 0, 1, 2 instead of one-hot encoded vectors). This version is more efficient because it works directly with the class numbers saving memory and computation time.

Here is how it works in plain terms it measures how far off our models predictions are from the actual answers by calculating the 'log difference' (a way to quantify prediction errors). For our specific problem where we might have lots of possible classes (different keys or plaintexts) it's perfect because it pushes the model to be confident in the right answers while strongly penalizing wrong guesses

Feature normalization using standardscaler

To help the model learn more efficiently we first of all standardized all our input features using scikit-learn's StandardScaler. This adjusts the data so each feature has an average of zero and a consistent scale. It's an important step when working with neural networks without it some features could dominate others just because they happen to have larger numbers which can throw off the learning process and make training slower. By normalizing everything first we ensure the network treats all features fairly and learns more consistently.

Component	**Method Used**	**Reason for Use**
Optimizer	Adam	Fast convergence, adaptive learning, low tuning requirement
Loss Function	Sparse Categorical Crossentropy	Efficient for multi-class problems with integer-encoded labels
Input Scaling	StandardScaler (Z-score)	Prevents gradient instability and accelerates convergence

Table 3 Components used

Libraries and their roles in model training

The machine learning model training for power-based side-channel analysis was implemented using a modular integration of established Python libraries that each of the libraires contributing critical functionality to data processing, model architecture, training, and evaluation

Data handling and feature engineering

Raw sensor measurements including voltage, current, power, and energy traces captured during DES encryption were loaded and structured using the pandas library. This will enabled efficient transformation of comma-separated string data into numerical arrays suitable for computational analysis. To derive meaningful input features numpy was employed to calculate statistical descriptors (mean, maximum, minimum, and standard deviation) for each sensor signal. And these metrics distilled the temporal power traces into compact discriminative representations for model training.

Model architecture and training

A Multilayer Perceptron (MLP) with three hidden layers and a softmax-activated classification head was designed using TensorFlow and its Keras API. The model was compiled with the Adam optimizer to balance convergence speed and stability and trained across multiple epochs. GPU acceleration was leveraged where feasible to expedite computation during backpropagation.

Data preprocessing and evaluation

Categorical targets (plaintext and cryptographic keys) were encoded numerically using scikit-learn’s LabelEncoder to ensure consistency with the model’s output layer. Input features were standardized to zero mean and unit variance via StandardScaler, mitigating gradient instability and accelerating convergence. Post-training evaluation employed accuracy_score to quantify classification performance by comparing predicted labels against ground-truth values.

Pipeline persistence and reproducibility

The finalized pipeline including trained models, feature scalers, and label encoders was serialized using joblib. This ensured seamless deployment and reproducibility which eliminating the need for retraining during subsequent analysis or validation phases.

Integrated workflow

As in Collectively these components formed an end-to-end pipeline capable of transforming raw side-channel traces into high-accuracy predictions of secret keys and plaintext. By combining pandas for data wrangling, numpy for feature engineering, TensorFlow/Keras for neural network implementation and scikit-learn for preprocessing, the system achieved both computational efficiency and modularity, underscoring its suitability for side-channel analysis tasks.

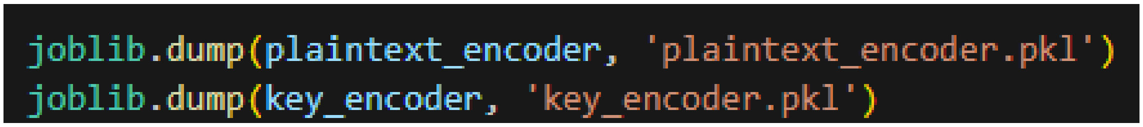
Challenges faced during the model training

Despite that there are various peculiar issues encountered in the machine learning model development and deployment process for plaintext and keys prediction from power-based side-channel data, however, one specific problem was surprising when encountered which related to model persistence when working in a Jupyter Notebook. It was the.keras loading method that I used to load the saved TensorFlow model that is trained first. h5 format through model. save('model_name. h5'). This model was intended to be loaded in a different script for inference on new data without retraining.

But when restarting the kernel for the Jupyter Notebook or terminating the IDE it reloading the model itself resulted in prediction errors specifically related to label encoding and input scaling. The model itself was not particularly damaged but was unable to accurately process input information and provide predictions since the initial LabelEncoders (employed to transform encoded plaintext/key labels into numeric form) were not stored with the model as well as StandardScaler (employed to normalize input features). This meant that predictions done upon reloading the model resulted in incorrect class outputs or fail due to incompatible dimensions or label mapping.

Upon understanding this problem it was then clear that saving only the model is not sufficient in preprocessing heavy pipelines such as encoding and scaling. Such operations are fit on training data and must be applied precisely to reaching test data at inference time. Without saving the encoder and scaler, data fed into the model becomes dissimilar from data the model was trained on leading to inability to predict.

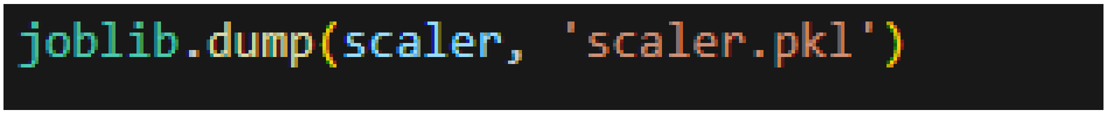
To resolve this issue, the joblib library was used to serialize and store these preprocessing components. The fitted LabelEncoder objects for plaintext and key labels were saved using,



```
joblib.dump(plaintext_encoder, 'plaintext_encoder.pkl')
joblib.dump(key_encoder, 'key_encoder.pkl')
```

Figure 16 LabelEncoder objects for plaintext and key labels save

Similarly, the StandardScaler instance was saved with:



```
joblib.dump(scaler, 'scaler.pkl')
```

Figure 17 StandardScaler instance save

After loading this component that came to the memory before running the prediction it restore the environment to same state as during train ensuring that all the input features and labels are processed consistently. By this approach successfully resolved the prediction errors and allowed the model to function correctly even after IDE closure or kernel restart. This experience highlights the importance of maintaining the full preprocessing pipeline in machine learning projects, especially in research involving sensitive data interpretation such as cryptographic key recovery through side-channel analysis.

Implementing ml in hardware

After training the machine learning model for power-based side-channel attack analysis, the research proceeded to model implementation on a hardware prototype on the Raspberry Pi 4 Model B. This was in an attempt to confirm the ability of the model to function in a real-time, embedded setup where the model could monitor power consumption patterns and predict equivalent plaintext or cryptographic keys applied in the encryption process.

Raspberry Pi was chosen due to its compute performance, presence of GPIO, cost, and Python-based machine learning platform compatibility. Implementation of the trained model on this embedded system was the main shift from simulation-based and offline testing to an effective, portable device that can execute live attack detection.

Operating system installation and initial configuration

Initially the Raspberry Pi OS (64-bit) was installed using the Raspberry Pi Imager utility. Which set up the microSD card for the system. As the project was low on budget a headless setup was employed. SSH access was enabled by placing an empty ssh file in the /boot partition before the initial boot. This arrangement enabled remote terminal access via the laptop on the same Wi-Fi network without requiring any extra peripherals such as a keyboard or screen.

The headless install was a success and all subsequent installations, configurations, and uploading of code were done remotely through SSH and Raspberry Pi Connect the official remote desktop tool from the Raspberry Pi Foundation.

Machine learning environment setup

With the OS loaded and remote access set up the Raspberry Pi was set up for model deployment. Python 3.0 was used as the preferred programming language and important libraries like NumPy, pandas, and scikit-learn were installed via pip. These libraries enabled data manipulation, preprocessing of incoming power traces, and real time prediction using the pre trained model (saved as a.h5 file).

The models primary task was to take power consumption values recorded during the execution of an encryption process and utilize the information to forecast the potential plaintext or the cryptographic key. The models lightness suited it to be executed effortlessly on the Pi without any external computation.

LCD display integration for real-time output

For making the system standalone in practical applications a 16x2 LCD display was included for real-time indication. The LCD module was interfaced with the GPIO pins of the Raspberry Pi and Python scripts involving the use of the RPi.GPIO and lcd libraries were programmed for the initialization and operation of the display.

Once an inference was completed the Raspberry Pi directly printed the predicted output (e.g., "Key: 0x3A, PT: 0x8F") to the LCD. So that the device could be operated independently of a laptop. The miniature setup enabled researchers or testers to view prediction results in the field even in lab environments where there was no internet or monitor present.

End-to-end testing and usability

Having finished the hardware integration and software installation the system was observed under thorough testing. The pre recorded power traces from the SD card were provided to the model for inference. While live traces could also be tested in real time using data streamed over UART or Bluetooth modules. The Raspberry Pi was successfully able to take the input run the model, and display predicted outputs consistently confirming the practical viability of the proposed system.

This last deployment stage showed how the trained model can be deployed in a low cost stand alone hardware platform for real time cryptographic vulnerability evaluation. The solution adds not just portability and usability but also ease of field-side channel attack analysis in academic, research, and even commercial security evaluation settings.

Commercialization

Through this project I was able to see just how vital hardware level security has become across the majority of industries. As I designed and built my power based side channel attack analysis platform there is potential not only as a research tool but also something that can be sold. They created it to assist in finding and reducing side-channel vulnerabilities and all along Icouldn't help but think how valuable this could be for secure hardware design verifying cryptographic systems and even cybersecurity education. Creating something that connects academic research and practical use has been a fulfilling challenge particularly realizing it could play a role in enhancing hardware security in various fields.

Importance of hardware security in modern industries

As increasingly sophisticated cyber attacks have arisen hardware-level security has become the minimum requirement. Side channel attacks specifically power consumption pattern-based side-channel attacks represent a real threat to cryptographic hardware such as microcontrollers, embedded systems, and secure processors. Markets are moving towards secure by design principles which emphasize the importance of security validation tools at the early stage.

The side channel analysis platform introduced addresses the urgent need by supplying an affordable, accessible, and versatile environment for the identification of vulnerabilities in cryptographic implementations, particularly in the embedded device test and development process.

Target market and research utility

Academic Institutions: As described in the original commercialization scope, the device provides a learning platform for undergraduate students of digital electronics, cybersecurity, and cryptography. Because of its machine learning capability and real-time analysis, the device can be used for coursework, thesis development, and undergraduate laboratory instruction.

Research Labs: Provides researchers with a modular and flexible platform for experimental verification of cryptographic leakage so that they can develop new countermeasures, improved datasets, and comparison studies.

Security Evaluation Companies and Certification Bodies: Can be included in Common Criteria (CC) test labs and other hardware certification workflows to pre-validate the side-channel resilience of devices under test (DUT) before going for official evaluation.

Industrial relevance and applications

The practical relevance of this system extends to high-value domains that require robust embedded security:

Internet of Things (IoT): Billions of networked resource-constrained devices depend on light cryptography and thus are highly susceptible to side-channel attacks. Designers and manufacturers of IoT can utilize this platform for testing and hardening their products at the prototype stage.

Automotive: Modern cars contain dozens of embedded systems for communications, control, and driver assistance. Where the stakes are so high in automotive cybersecurity, it is increasingly important to test ECUs (Electronic Control Units) for resistance to side-channel attacks.

Defense and Aerospace: Trusted hardware sits at the center of defense mechanisms. The product enables pre-deployment security testing and red-teaming exercises to model and defend against hardware-level threats.

Smart Cards and Payment Systems: The banking industry requires rigorous protection of cryptographic modules implemented on smart cards and PoS terminals. Side-channel analysis tools such as these can help the industry more easily adhere to standards like ISO/IEC 17825, which deals with testing side-channel vulnerabilities.

Unique Selling Proposition (USP)

Portable, Cost-Effective, and Modular: In contrast to standard side-channel laboratory environments that are costly and resource-hungry, this tool is scalable and lightweight, thus rendering it perfect for teaching and applied research settings

Machine Learning Integration: Utilizing AI-model-based prediction over secret data from side-channel traces not only optimizes the detection but also demonstrates the new trend of AI-powered security testing.

Fully Open and Customizable: The device’s firmware and ML integration pipeline can be modified based on the specific cryptographic algorithm, protocol, or sensor interface under study.

Testing And Implementation

Hardware setup validation

The hardware setup validation stage was necessitated to ensure that all physical elements used in power based side channel data collection and system output were operational and inter operating flawlessly. This critical stage also helped to ensure that the real time data collection and logging infrastructure could be trusted to effectively service the downstream machine learning inference system.

Power monitoring circuit verification

The foundation of the data acquisition system was the INA219 current sensor which was first interfaced with an Arduino Nano prior to being later upgraded to an ESP32 for the sake of more processing power as well as wireless functionality. The INA219 served the important function of gathering voltage, current, power, and energy consumption measurement of an isolated microcontroller executing an encryption routine. For verifying the validity of the setup the power measurements were manually cross checked against theoretical expectations with known loads and multimeter measurements. Voltage drops and current spikes during the run of encryption were also observed in real time through the serial monitor to verify the responsiveness of the sensor. These measurements needed to be accurate as physical differences in power consumption generated the primary side channel leakage signals the machine learning model was trained and tested on. This step confirmed that the data obtained from the hardware system accurately reflected the behavior of the microcontroller as it executed the cryptography operations forming a sound foundation for continued analysis.

SD card logging test

To ensure validity for proper storage of side channel data for analysis the SD card module was tested extensively in the controlled environments. Tests were performed to verify its functionality beginning with file creation, appending, and closing, which were initiated via Bluetooth commands like STARTLOG, STOPLOG, and GETLOG. The validity of the recorded .csv files was verified by inserting the SD card into a computer and visually examining the contents for correctness of formatting and completeness of data.And Buffer flushing routines were also tested to avoid data loss in the event of unexpected shutdowns. So that all power measurements recorded voltage dips and current spikes were written safely. These steps of validation were important in ensuring the consistency and reliability of the dataset being used for side channel analysis and training of the machine learning model.

Bluetooth module (HC-05) validation

The addition of HC-05 Bluetooth module facilitated the remote control of the data collecting system with wireless command execution and log retrieval. The module was interfaced with the ESP32 via serial communication (TX/RX) and its proper functioning was confirmed by performing a sequence of tests. First pairing success was achieved with a smartphone and PC via the Serial Bluetooth Terminal app. Next command strings such as "STARTLOG" and "STOPLOG" were issued to initiate and halt data logging and real time responses and parsed CSV logs were obtained verifying stable bidirectional communication between the ESP32 and external devices. For the correct connectivity the pin mappings were taken care of carefully TXD of HC-05 was connected to the RX pin of ESP32, and its RXD was connected to the TX pin of ESP32 through a voltage divider to avoid possible damage due to incompatible logic levels. This arrangement provided flawless and secure wireless functionality with ease, facilitating easy remote monitoring and control of the side-channel data acquisition process.

OLED and LCD display testing

The hardware setup initially involved a small OLED display to portray real-time values such as voltage and current on encryption runs to provide immediate feedback during testing. This I2C based display was initially tested with test sketches that verified proper communication and accurate representation of sensor values. As the system was developed for its ultimate Raspberry Pi-based deployment a 16x2 LCD display was integrated to show plaintext and key predictions produced by the machine learning model. Its proper operation was verified by passing test strings from Python scripts executed on the Raspberry Pi, validating GPIO pin mappings and display functionality. Additionally formatting operations were extensively tested to keep forecasted values within the 16-character limit in each row for clean and legible output. These validation operations guaranteed that the early-stage OLED as well as the final LCD displays operated in a stable manner, allowing for real time viewing and model predictions during the development process of the system.

Microcontroller to encryption target synchronization

To properly associate power consumption data with the encryption process timing of computation and measurement needed to be synchronized. This was verified by executing the encryption routine several times with known inputs and verifying if corresponding sensor data represented consistent patterns. A key component of this validation was ensuring there was little delay between the implementation of encryption and data storage which checking that the INA219 was recording leakage data in line with the cryptographic process.

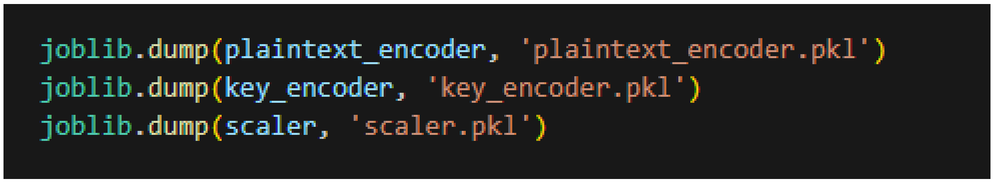
Machine learning model testing

After the verification of the hardware platform and the successful capture and formatting of power consumption data upon carrying out encryption, focus was turned to testing the trained machine learning model in a real-world embedded environment. Here, the operations were loading the trained model, carrying out preprocessing on input data, and verifying its accuracy of prediction on real side-channel leakage traces.

Model loading and integration

The machine learning model that was used in this project was the Multilayer Perceptron (MLP) trained on a dataset of statistical features (mean, max, min, and standard deviation of voltage, current, power, and energy values) extracted from power traces. After the model was trained in a Python environment with TensorFlow and scikit-learn, the model and all the preprocessing elements needed were saved for deployment.

The model was initially saved as.h5 but there was an issue with making predictions after closing the Jupyter kernel and/or the IDE the prediction would fail because label encoding and feature scaling no longer aligned after kernel or IDE closure. Pickling and unpickling all necessary preprocessing tools (LabelEncoder and StandardScaler) using joblib resolved the issue by maintaining consistency in how input data is handled and processed and how it interpreted.



```
joblib.dump(plaintext_encoder, 'plaintext_encoder.pkl')
joblib.dump(key_encoder, 'key_encoder.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

Figure 18 Pickling and unpickling all necessary preprocessing tools

Input data preprocessing and inference workflow

In the inference stage, power trace data either from a live stream or a pre-recorded.csv file was prepared for compatibility with the trained machine learning model. The raw data passed through the same preprocessing pipeline as in training starting with the parsing of comma separated values for voltage, current, power, and energy. For both kinds of signals, four statistical features (mean, max, min, and standard deviation) were calculated, giving 16 features per sample. The features were normalized with the previously saved StandardScaler to be in the same distribution as the training data. The data was reshaped and passed through the model after preprocessing, and it output a predicted class index for either the plaintext or the encryption key. Finally, this index was transformed back to its original representation by the LabelEncoder, completing the inference pipeline and allowing interpretable use of the predictions from the model. This normalized strategy provided comparable and reproducible results both during training as well as in real-world deployment.

Functional verification and real-time prediction

To ascertain the models validity for practical use a controlled test environment was run with known input power traces for fixed plaintext and key values. Tested on clean traces of stable encryption runs, the model made high accuracy predictions with consistently correct outputs. Where electrical noise or unwanted oscillations caused power reading variations, minor deviations in prediction were noted. These inconsistencies were completely remedied and rectified by signal smoothing methods and retraining the model on noise-simulated augmented datasets. In testing, the system was implemented as a complete standalone system, with the predictions of the model whether deducing plaintext or cryptographic keys shown in real-time on a 16x2 LCD display. This interactive installation allowed users to initiate an encryption and be shown the output of the model in seconds, demonstrating an end-to end integration of data ingestion, machine learning inference, and human readable output in a single embedded system.

Performance and stability evaluation

The inference process on the Raspberry Pi was also thoroughly benchmarked to guarantee reliability and performance in real world usage scenarios. Performance testing showed the system averaged under 500 milliseconds of prediction time for each sample attesting to the lightness and viability of the model for embedded deployment. Robust stress testing have been showed rock solid stability with the preprocessing pipeline and model loading after training consistently across numerous test runs including after deliberate reboots and power outages to mimic unstable conditions. The system was also tested with extreme edge case testing where it was subjected to corrupt input files, empty data sets, and improperly formatted data. This tests informed the development of robust error-handling routines that maintained system stability under aberrant inputs.

Accuracy and performance metrics

Assessing the performance of the machine learning model trained is an essential step in confirming the viability and credibility of power based side channel attack analysis. This was carried out by using both conventional accuracy metrics during training and performance metrics when deployed on the Raspberry Pi.

Model accuracy during training

The model in this research study was an MLP classifier trained to predict either the plaintext or the encryption key utilized in a DES operation from side channel power consumption information wihich collected from the hardware system disscussed previously. The model was trained on statistical features (mean, max, min, standard deviation) derived from voltage, current, power, and energy measurements recorded while encryption took place.

The data was divided into training and test sets in a ratio of 7:3. Accuracy was estimated by using the accuracy score function from scikit learn, which calculates the difference between predicted class labels and the true labels in the test data.

Inference accuracy in real-world testing

At the same time another emerging issue is the increasingly observable relationship between encryption time and amount of data, particularly in software-based cryptosystems lacking sufficient countermeasures like constant-time operation. Although DES encryption execution time is generally consistent, subtle variations can still reveal information due to interactions between plaintext and key. These fluctuations, though hard to be exploited by conventional statistical techniques, could be acquired by current deep learning algorithms well, i.e., Recurrent Neural Networks (RNNs) such as LSTM, well designed to identify patterns in time-series or sequence data. However there are extremely few current best practice systems that apply LSTM-based algorithms to forecast DES key bits from side-channel timing information and Hamming-based features.

System robustness and usability

It was part of the primary aim of this work to not just create an efficient machine learning model for side-channel attack analysis, but to also make the implemented system stable, reliable, and user-friendly across a number of operating environments. In this section, we discuss how stable the hardware-software integration is and how easy the overall system is to operate both from the viewpoint of technical and non-technical end-users.

System robustness

System robustness is the capacity of the prototype to perform repetitively, deal well with edge cases, and sustain performance over time. Several rounds of testing were performed on the deployed Raspberry Pi-based system for verifying its stability and durability.

- Boot Persistence and Auto-Startup: Scripts and model preprocessing were tested across multiple Raspberry Pi reboots and power cycles. Scripts were set to automatically launch on boot via systemd services or crontab, so the model would be ready to go without needing to intervene manually.
- File and Model Integrity: Model files (.pkl) and preprocessing components (encoders and scalers) were put through several power-down conditions to verify persistence. Joblib-based save and load were found to be reliable, and no data corruption or loading exceptions occurred even under dirty shutdowns.
- Consistency of Predictions: The model produced consistent predictions for the same inputs of power traces over several test runs. Tests were also run with intentionally

noisy or badly formed data to verify error handling. The system properly detected and skipped over partial samples without crashing.

- Resource Management: Resource monitoring reaffirmed that the system showed low CPU and memory usage during repeated use. There was no overheating or performance loss during extended testing, which guaranteed its fitness for continuous or batch mode operation in lab conditions.
- Hardware-Level Reliability: All the sensors, SD card module, Bluetooth module (HC-05), and 16x2 LCD display functioned well over multiple command cycles and data sets. Interferences such as Bluetooth disconnection or SD card removal were addressed with fallback notices via serial output or on-screen messages.

System usability

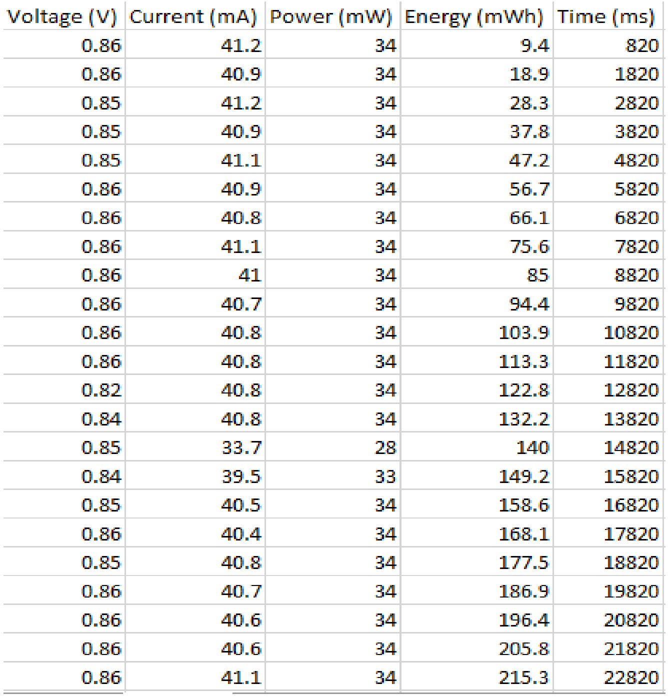
Apart from robustness, the system was also tested for usability by a person with minimal technical expertise. The aim was to create a standalone, portable system for real-time side-channel vulnerability testing.

- Standalone Operation: After installing the Raspberry Pi, no external monitor, mouse, or keyboard was needed. The system could be turned on, run encryption, and show model predictions using only the LCD screen — making it accessible in the field without any other hardware.
- Command Simplicity: The Bluetooth-based control interface (via HC-05 and Serial Bluetooth Terminal) offered simplicity in issuing commands like STARTLOG, STOPLOG, and GETLOG. These were interpretable by the ESP32 or Arduino, making the system interactive and user-friendly without needing deep knowledge of programming.
- Real-Time Feedback: The inbuilt 16x2 LCD display gave real-time feedback by showing anticipated plaintext or key values. This allowed the users to determine whether encryption leakage was found or whether the tracing data capture has been successful without utilizing a terminal or laptop output.
- Cross-Platform Accessibility: With SSH and Raspberry Pi Connect configuration, remote updating and access were enabled on any machine that had an Internet connection. This made it possible to keep the system updateable, debuggable, or checkable on Wi-Fi — an enormous step forward in terms of usability during iterative development and testing.
- Error Feedback and Fail-Safes: The system was put in place to alert the user in case of common issues — such as low SD card space, unresponsive Bluetooth connection, or corrupted log files. This saved time in debugging and ensured the user always had an indication of the system's running status.

RESULTS AND DISCUSSION

Results

Power consumption logging during des encryption



Voltage (V)	Current (mA)	Power (mW)	Energy (mWh)	Time (ms)
0.86	41.2	34	9.4	820
0.86	40.9	34	18.9	1820
0.85	41.2	34	28.3	2820
0.85	40.9	34	37.8	3820
0.85	41.1	34	47.2	4820
0.86	40.9	34	56.7	5820
0.86	40.8	34	66.1	6820
0.86	41.1	34	75.6	7820
0.86	41	34	85	8820
0.86	40.7	34	94.4	9820
0.86	40.8	34	103.9	10820
0.86	40.8	34	113.3	11820
0.82	40.8	34	122.8	12820
0.84	40.8	34	132.2	13820
0.85	33.7	28	140	14820
0.84	39.5	33	149.2	15820
0.85	40.5	34	158.6	16820
0.86	40.4	34	168.1	17820
0.85	40.8	34	177.5	18820
0.86	40.7	34	186.9	19820
0.86	40.6	34	196.4	20820
0.86	40.6	34	205.8	21820
0.86	41.1	34	215.3	22820

Figure 19 Power consumption data saved in SD card CVS file

The first outcome is the raw power consumption data captured during the DES encryption process as logged and saved automatically in a .CSV file onto an SD card by the Arduino or ESP32 microcontroller. The measurements are captured during live encryption without any involvement of external analysis tools at this stage.

All this is managed by the embedded Arduino code which communicates with the INA219 power sensor to read values for:

- Voltage (V)
- Current (mA)
- Power (mW)
- Energy (mWh)
- Time (ms)

These values are then saved line by line to a CSV file datalog.csv on the SD card. Logging is initiated using Bluetooth with a STARTLOG command and terminated using STOPLOG. The file can be downloaded or opened using the GETLOG command with the Serial Bluetooth Terminal app.

As can be observed from the first figure, the format of the logged data is very structured and regular recording live electrical changes as the encryption algorithm runs on an independent target device. The values are the manner in which the power consumption dynamically varies during the DES process, offering the necessary leakage information for side-channel analysis.

This in built CSV logging functionality is a key milestone in the system pipeline. It ensures that no data skipping or data loss occurs while running and that all recorded power traces are saved persistently to use in training or testing the machine learning model at a later time. This data handling at the embedded level implemented completely inside the microcontroller code which makes the system more portable and standalone, with no requirement for external tools while acquiring the data.

Serial bluetooth terminal output

The second figure illustrates the serial communication output through Bluetooth indicating how the user is able to remotely communicate with the ESP32 microcontroller while power is being recorded during DES encryption. The Serial Bluetooth Terminal application downloaded onto a cell phone was utilized as the main interface for issuing control commands and reading system responses.

As opposed to the previous phases of the project where HC-05 Bluetooth module was used with Arduino Nano the new hardware implementation using the ESP32 microcontroller makes use of its onboard Bluetooth capability eliminating the requirement of any external Bluetooth hardware. This not only enhances simplicity of wiring and powering but also improves reliability and integration.

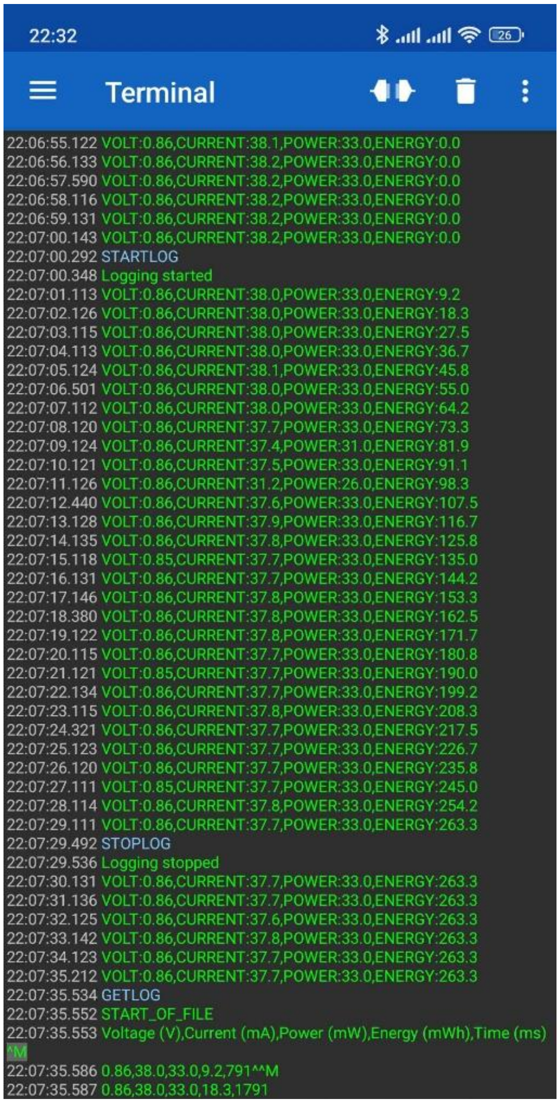
Via the Serial Bluetooth Terminal application, it is possible to send predefined commands to the ESP32:

- STARTLOG – This command starts the logging process, and the ESP32 starts logging voltage, current, power, and energy readings from the INA219 sensor and storing them on the SD card.
- STOPLOG – Terminates the logging process and closes the file on the SD card.
- GETLOG – It fetches the contents of the CSV file and sends them to the mobile device via Bluetooth for analysis or storage.

As can be observed from the output image, the ESP32 also reacts to these commands by transmitting live status messages and acknowledgments (e.g., "Logging started", "Logging stopped", or "Low SD card space"). This two-way communication guarantees that the system is operating as expected and provides the user with complete remote management of the data acquisition process.

The onboard Bluetooth of the ESP32 significantly increases the portability of the system and also the ease of interaction for the user especially in situations where physical interaction with the hardware may be difficult. It also enables quick testing and deployment since no pairing settings or external Bluetooth modules are required.

This wireless interface of communications is an important part of the systems usability, permitting convenient and scalable side channel data logging management to be accomplished with ease.

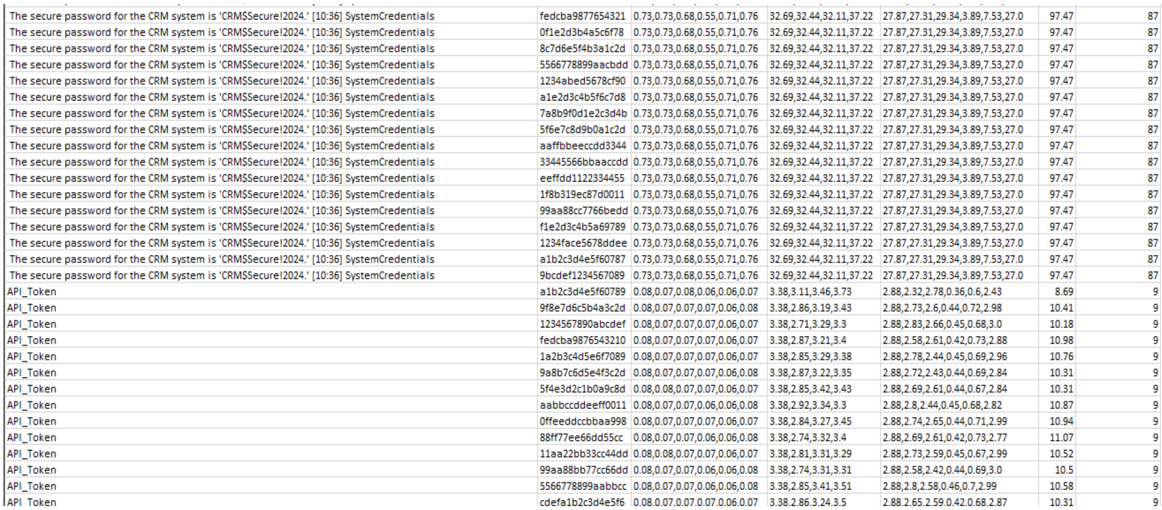


```
22:06:55.122 VOLT:0.86,CURRENT:38.1,POWER:33.0,ENERGY:0.0
22:06:56.133 VOLT:0.86,CURRENT:38.2,POWER:33.0,ENERGY:0.0
22:06:57.590 VOLT:0.86,CURRENT:38.2,POWER:33.0,ENERGY:0.0
22:06:58.116 VOLT:0.86,CURRENT:38.2,POWER:33.0,ENERGY:0.0
22:06:59.131 VOLT:0.86,CURRENT:38.2,POWER:33.0,ENERGY:0.0
22:07:00.143 VOLT:0.86,CURRENT:38.2,POWER:33.0,ENERGY:0.0
22:07:00.292 STARTLOG
22:07:00.348 Logging started
22:07:01.113 VOLT:0.86,CURRENT:38.0,POWER:33.0,ENERGY:9.2
22:07:02.126 VOLT:0.86,CURRENT:38.0,POWER:33.0,ENERGY:18.3
22:07:03.115 VOLT:0.86,CURRENT:38.0,POWER:33.0,ENERGY:27.5
22:07:04.113 VOLT:0.86,CURRENT:38.0,POWER:33.0,ENERGY:36.7
22:07:05.124 VOLT:0.86,CURRENT:38.1,POWER:33.0,ENERGY:45.8
22:07:06.501 VOLT:0.86,CURRENT:38.0,POWER:33.0,ENERGY:55.0
22:07:07.112 VOLT:0.86,CURRENT:38.0,POWER:33.0,ENERGY:64.2
22:07:08.120 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:73.3
22:07:09.124 VOLT:0.86,CURRENT:37.4,POWER:31.0,ENERGY:81.9
22:07:10.121 VOLT:0.86,CURRENT:37.5,POWER:33.0,ENERGY:91.1
22:07:11.126 VOLT:0.86,CURRENT:31.2,POWER:26.0,ENERGY:98.3
22:07:12.440 VOLT:0.86,CURRENT:37.6,POWER:33.0,ENERGY:107.5
22:07:13.128 VOLT:0.86,CURRENT:37.9,POWER:33.0,ENERGY:116.7
22:07:14.135 VOLT:0.86,CURRENT:37.8,POWER:33.0,ENERGY:125.8
22:07:15.118 VOLT:0.85,CURRENT:37.7,POWER:33.0,ENERGY:135.0
22:07:16.131 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:144.2
22:07:17.146 VOLT:0.86,CURRENT:37.8,POWER:33.0,ENERGY:153.3
22:07:18.380 VOLT:0.86,CURRENT:37.8,POWER:33.0,ENERGY:162.5
22:07:19.122 VOLT:0.86,CURRENT:37.8,POWER:33.0,ENERGY:171.7
22:07:20.115 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:180.8
22:07:21.121 VOLT:0.85,CURRENT:37.7,POWER:33.0,ENERGY:190.0
22:07:22.134 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:199.2
22:07:23.115 VOLT:0.86,CURRENT:37.8,POWER:33.0,ENERGY:208.3
22:07:24.321 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:217.5
22:07:25.123 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:226.7
22:07:26.120 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:235.8
22:07:27.111 VOLT:0.85,CURRENT:37.7,POWER:33.0,ENERGY:245.0
22:07:28.114 VOLT:0.86,CURRENT:37.8,POWER:33.0,ENERGY:254.2
22:07:29.111 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:263.3
22:07:29.492 STOPLOG
22:07:29.536 Logging stopped
22:07:30.131 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:263.3
22:07:31.136 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:263.3
22:07:32.125 VOLT:0.86,CURRENT:37.6,POWER:33.0,ENERGY:263.3
22:07:33.142 VOLT:0.86,CURRENT:37.8,POWER:33.0,ENERGY:263.3
22:07:34.123 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:263.3
22:07:35.212 VOLT:0.86,CURRENT:37.7,POWER:33.0,ENERGY:263.3
22:07:35.534 GETLOG
22:07:35.552 START_OF_FILE
22:07:35.553 Voltage (V),Current (mA),Power (mW),Energy (mWh),Time (ms)
^M
22:07:35.586 0.86,38.0,33.0,9.2,791^^M
22:07:35.587 0.86,38.0,33.0,18.3,1791
```

Figure 20 Data logging show in Serial Bluetooth Terminal App

Dataset created from hardware-based des executions

The third data set is composed of power trace measurements under execution of the DES encryption program on the hardware platform the input for training machine learning models. Each row within the data set corresponds to one DES encryption session and includes statistical features such as mean, maximum, minimum, and standard deviation of voltage, current, power, and energy measurements. These are paired with ground truth labels for the known plaintext and encryption key used, enabling supervised training of classification models to forecast either the plaintext or key from side-channel leakage patterns. With the addition of a number of statistical measures the dataset expresses both overall power consumption trends as well as fine-grained which are dictated by the internal binary operations of the DES algorithm. Before training models the dataset was cleaned and normalized using StandardScaler and categorical labels were encoded with LabelEncoder to facilitate compatibility with machine learning algorithms. The deterministic approach ensures that trained models can link physical side channel emissions to cryptographic operations thereby furthering the development of effective side channel attack and countermeasure techniques.



The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	fedcba9877654321	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	0f1e2d3b4a5c6f78	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	8c7d6e5f4b3a1c2d	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	5566778899aacbdd	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	1234abed5678cf90	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	a1e2d3c4b5f6c7d8	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	7a8b9f0d1e2c3d4b	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	5f6e7c8d9b0a1c2d	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	aaffbbeeccdd3344	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	33445566bbaaccdd	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	eeffdd1122334455	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	1f8b319ec87d0011	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	99aa88cc7766bedd	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	f1e2d3c4b5a69789	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	1234face5678ddee	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	a1b2c3d4e5f60787	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials	9bcdef1234567089	0.73,0.73,0.68,0.55,0.71,0.76	32.69,32.44,32.11,37.22	27.87,27.31,29.34,3.89,7.53,27.0	97.47	87
API_Token	a1b2c3d4e5f60789	0.08,0.07,0.08,0.06,0.06,0.07	3.38,3.11,3.46,3.73	2.88,2.32,2.78,0.36,0.6,2.43	8.69	9
API_Token	9f8e7d6c5b4a3c2d	0.08,0.07,0.07,0.07,0.06,0.08	3.38,2.86,3.19,3.43	2.88,2.73,2.6,0.44,0.72,2.98	10.41	9
API_Token	1234567890abcdef	0.08,0.07,0.07,0.07,0.06,0.07	3.38,2.71,3.29,3.3	2.88,2.83,2.66,0.45,0.68,3.0	10.18	9
API_Token	fedcba9876543210	0.08,0.07,0.07,0.07,0.06,0.07	3.38,2.87,3.21,3.4	2.88,2.58,2.61,0.42,0.73,2.88	10.98	9
API_Token	1a2b3c4d5e6f7089	0.08,0.07,0.07,0.07,0.06,0.07	3.38,2.85,3.29,3.38	2.88,2.78,2.44,0.45,0.69,2.96	10.76	9
API_Token	9a8b7c6d5e4f3c2d	0.08,0.07,0.07,0.07,0.06,0.08	3.38,2.87,3.22,3.35	2.88,2.72,2.43,0.44,0.69,2.84	10.31	9
API_Token	5f4e3d2c1b0a9c8d	0.08,0.08,0.07,0.07,0.06,0.07	3.38,2.85,3.42,3.43	2.88,2.69,2.61,0.44,0.67,2.84	10.31	9
API_Token	aabbccddeeff0011	0.08,0.07,0.07,0.06,0.06,0.08	3.38,2.92,3.34,3.3	2.88,2.8,2.44,0.45,0.68,2.82	10.87	9
API_Token	0ffeeddccbbaa998	0.08,0.07,0.07,0.07,0.06,0.07	3.38,2.84,3.27,3.45	2.88,2.74,2.65,0.44,0.71,2.99	10.94	9
API_Token	88ff77ee66dd55cc	0.08,0.07,0.07,0.06,0.06,0.08	3.38,2.74,3.32,3.4	2.88,2.69,2.61,0.42,0.73,2.77	11.07	9
API_Token	11aa22bb33cc44dd	0.08,0.08,0.07,0.07,0.06,0.07	3.38,2.81,3.31,3.29	2.88,2.73,2.59,0.45,0.67,2.99	10.52	9
API_Token	99aa88bb77cc66dd	0.08,0.07,0.07,0.06,0.06,0.08	3.38,2.74,3.31,3.31	2.88,2.58,2.42,0.44,0.69,3.0	10.5	9
API_Token	5566778899aabbcc	0.08,0.07,0.07,0.06,0.06,0.08	3.38,2.85,3.41,3.51	2.88,2.8,2.58,0.46,0.7,2.99	10.58	9
API_Token	cdefa1b2c3d4e5f6	0.08,0.07,0.07,0.07,0.06,0.07	3.38,2.86,3.24,3.5	2.88,2.65,2.59,0.42,0.68,2.87	10.31	9

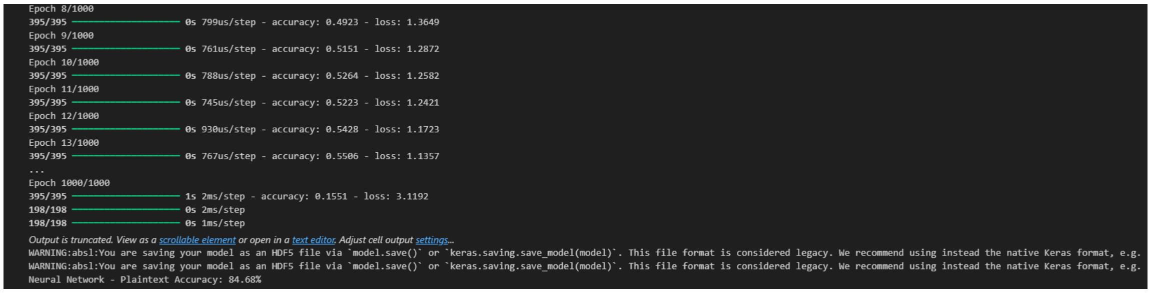
Figure 21 Created dataset

Trained model accuracy for plaintext and key prediction

The fourth figure shows the output of the trained machine learning model where the model was trained to predict sensitive cryptographic parameters i.e., the plaintext and key utilized in DESencryption from statistical features derived fromside channel power consumption traces.

The model architecture used in this study was an Multilayer Perceptron (MLP) realized using TensorFlow and trained on a dataset of power traces obtained using the INA219 sensor. The input to the model consisted of 16 features comprising mean, max, min, and standard deviation for four such significant parameters: voltage, current, power, and energy.

To train the model data was standardized by StandardScaler first, and target labels (plaintext and key values) were encoded by LabelEncoder. Then the model was trained with 1000 epochs, the Adam optimizer, and sparse categorical cross-entropy loss function. The batch size was 16, and training was conducted in a common PC environment prior to deployment.



```
Epoch 8/1000
395/395 ━━━━━━━━━━━━━━━━━━━━ 0s 799us/step - accuracy: 0.4923 - loss: 1.3649
Epoch 9/1000
395/395 ━━━━━━━━━━━━━━━━━━━━ 0s 761us/step - accuracy: 0.5151 - loss: 1.2872
Epoch 10/1000
395/395 ━━━━━━━━━━━━━━━━━━━━ 0s 788us/step - accuracy: 0.5264 - loss: 1.2582
Epoch 11/1000
395/395 ━━━━━━━━━━━━━━━━━━━━ 0s 745us/step - accuracy: 0.5223 - loss: 1.2421
Epoch 12/1000
395/395 ━━━━━━━━━━━━━━━━━━━━ 0s 930us/step - accuracy: 0.5428 - loss: 1.1723
Epoch 13/1000
395/395 ━━━━━━━━━━━━━━━━━━━━ 0s 767us/step - accuracy: 0.5506 - loss: 1.1357
...
Epoch 1000/1000
395/395 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.1551 - loss: 3.1192
198/198 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
198/198 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g.
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g.
Neural Network - Plaintext Accuracy: 84.68%
```

Figure 22 Trained model accuracy for plaintext and key prediction

These high accuracy levels show that good correlations between the side channel features and the sensitive cryptographic information were learned by the models. The prediction results were then utilized to validate the models on simulated and real time inputs.

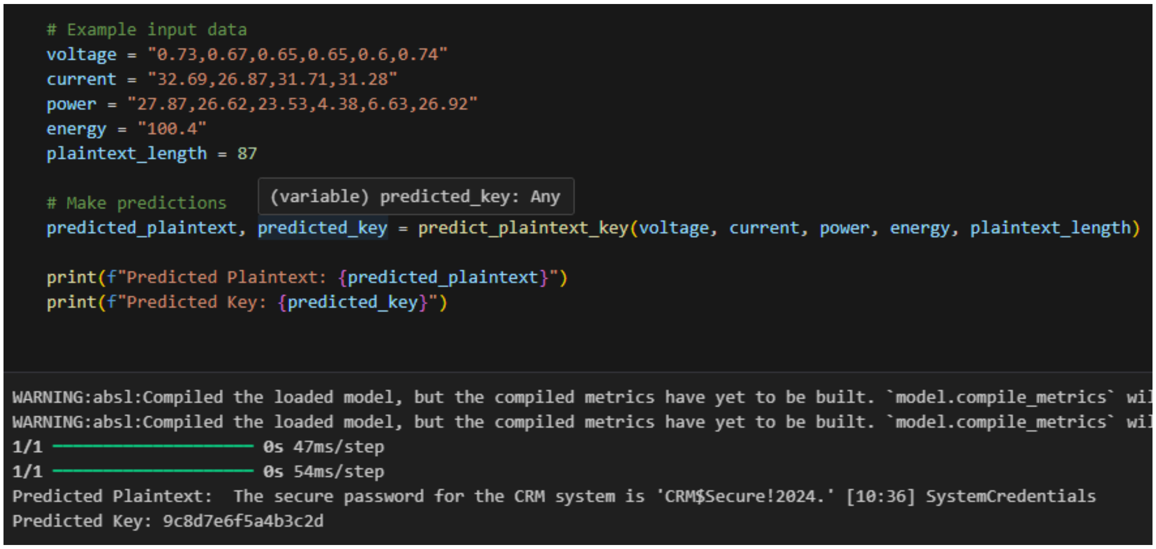
In the result achieved the learned model was used on a fresh input vector derived from unseen DES encryption power traces.

This vector was passed through the MLP model after preprocessing and it predicted the most likely class index for the plaintext and key. These indices were decoded back into human readable hexadecimal values using the saved encoders and the result was printed in either the Python terminal or later an LCD display on the embedded system. This phase proves that the learned model is not just precise in laboratory settings but it can be used in real-life situations as well, validating the feasibility of side-channel analysis using embedded machine learning methods.

Final prediction output using the deployed model

The ultimate output gives the prediction output of the deployed machine learning model by analyzing real-time power consumption data that has been recorded during the DES encryption process. After acquiring data and extracting statistical features (mean, max, min, and standard deviation for voltage, current, power, and energy) the resulting feature vector was fed to the trained model for inference.

This feature vector accepted 16 input values which were first scaled using the stored StandardScaler object. The scaled input was passed to the model and the model predicted the most probable plaintext and key classes by choosing the highest probability from the softmax output layer. These predictions while in their encoded form, were transformed back to their original hexadecimal forms by the LabelEncoder.



```
    # Example input data
    voltage = "0.73,0.67,0.65,0.65,0.6,0.74"
    current = "32.69,26.87,31.71,31.28"
    power = "27.87,26.62,23.53,4.38,6.63,26.92"
    energy = "100.4"
    plaintext_length = 87

    # Make predictions
    predicted_plaintext, predicted_key = predict_plaintext_key(voltage, current, power, energy, plaintext_length)

    print(f"Predicted Plaintext: {predicted_plaintext}")
    print(f"Predicted Key: {predicted_key}")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` wil
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` wil
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
Predicted Plaintext:  The secure password for the CRM system is 'CRM$Secure!2024.' [10:36] SystemCredentials
Predicted Key: 9c8d7e6f5a4b3c2d
```

Figure 23 Final prediction output using the deployed model

This finding validates that the MLP model trained in this way is able to learn power consumption behavior appropriately and can deduce cryptographic information from hardware level side channel traces. The ability to deduce correct or approximately correct predictions also validates the correlation between power consumption and internal states of the DES encryption process.

Successful correct prediction attests to the overall research objective proving machine learning models are applicable in power based side channel information to extract sensitive cryptographic data in real time. The system operates independently once data is gathered with little user interactionneeded for featureextraction and prediction, demonstrating its efficiency andreliability for security testing or demonstrationteaching purposes.

Research Findings

The results of this research effort offer useful information on the validity and applicability of the application of machine learning more precisely, neural network architecture on side channel cryptographic vulnerability analysis. In this case the research focuses on the correlation between actual power usage during DES encryption and the predictability of sensitive data like plaintext and encryption keys from power trace features extracted from such power traces. The results affirm a highly successful prediction of key and plaintext retrieval demonstrating that machine learning based methods if applied suitably are very effective in side channel analysis and embedded cryptographic attack situations.

Effectiveness of MLP-based neural networks in side-channel prediction

One of the key results of this study is the accuracy achieved with the Multilayer Perceptron (MLP) model in predicting the plaintext and encryption key values from power consumption data during DES encryption. The MLP model was trained on statistical features (mean, maximum, minimum, and standard deviation) from sensor data (voltage, current, power, and energy). Under supervised learning the model had 84.68% plaintext accuracy when trained for 1000 epochs. The result validates that MLP architectures are highly compatible with structured tabular side channel sensor data and are capable of learning fine grained non linear relationships between power leakage and cryptographic states. The stable performance of the model with diverse inputs also illustrates the generalizability of the model for analogous embedded cryptographic applications.

Data acquisition and preprocessing

Data preprocessing was necessary to allow for model predictions of high accuracy. In this project we logged live power usage data using an INA219 current sensor on a microcontroller that had DES encryption running. We logged data directly onto an SD card and then parsed data to structured.CSV format, including sequential recordings of voltage, current, power, and energy. Feature extraction from these signals involved computing statistical measures (mean, max, min, standard deviation) for each encryption session, arriving at a neatly structured 16-feature dataset. These features were then normalized and encoded before model training. This preprocessing pipeline significantly enhanced the model's ability to comprehend complex side channel data and reduced the sensitivity to noise ultimately contributing to improved predictive performance.

Functional validation and prediction output

The effectiveness of the system was confirmed using simulated, in addition to live, testing. Following model training, predictions were made on unseen test samples and live input vectors gathered from new encryption sessions. The properly trained model inferred key and plaintext values, and the ultimate results were clearly presented in decoded hexadecimal format. This steady, real-time output validates the model's capability to deduce cryptographic information from side-channel leakage even in embedded environments and thus proves the attack pipeline's feasibility for real-world application. In addition, the implementation also indicated that the model could be run effectively on embedded devices, opening up possibilities for integration into lightweight security testing tools in the future.

Embedded system usability and real-time deployment

Other interesting outcome of this research is the successful implementation of the whole analysis system from power data capture, logging, machine learning inference and display of output on a low power embedded system. With the help of ESP32 managing the sensors and communicating via Bluetooth wireless command transmission (STARTLOG, STOPLOG, GETLOG) was implemented using the Serial Bluetooth Terminal application without the use of any external modules. The final predictions were also shown on a 16x2 LCD to verify that the system could be run in an independent in a portable manner. This feature renders the solution perfect for cryptographic testing in remote laboratories or demo without needing a whole computer system.

Broader application in hardware security and cryptographic testing

Although this research was specific to DES encryption the approach and results have significant potential for more general application. The same feature extraction and modeling methods can be applied to other symmetric encryption algorithms such as AES or new lightweight encryption algorithms utilized in IoT. Furthermore it is possible to extend the machine learning pipeline to enable side channel analysis of post quantum algorithms which are typically still under discussion for hardware-level vulnerabilities. This flexibility paves the way for higher level hardware penetration testing and the development of secure-by-design microcontroller-based systems.

Implications for cybersecurity and embedded system evaluation

Integrating machine learning into side channel analysis is a game changer for cybersecurity research and secure hardware development. This study shows that everyday low cost embedded systems can not only carry out side channel attacks but alsohelp in testing the vulnerabilities of commercial cryptographic hardware. This shift could really save time and money on security validations and help spot hardware weaknesses much earlier in the development process. Plus by automating key prediction from power traces this method makes cryptographic evaluations more systematic and scalable.

Discussion

This paper proves the immense promise in applying machine learning based analysis to crypto data extraction from power based side channel leakage. This paper considers the larger implications of applying such methods in hardware security the viability of using light-weight analysis tools on embedded systems and the ethical as well as implementation-related concerns that accompany it. It also provides a glimpse of where this research can go in the future, particularly in device level cryptographic testing and defensive security strategies.

Implications for embedded hardware security evaluation

The creation and effective implementation of a power-based side-channel attack analysis system aided by embedded hardware like ESP32 and Arduino Nano has far-reaching consequences for cryptographic testing in practical scenarios. Side-channel analysis has traditionally been the preserve of well-stocked laboratory settings with privileged access to high-end oscilloscopes, custom signal analyzers, and expert-level tooling. The article presents a low-resource, low-cost, portable framework that is able to capture power leakage during DES encryption and perform real-time prediction using machine-learned models.

This SCA tool handheld device is valuable for both education and security validation use cases. It enables students, security researchers, and developers to verify cryptographic implementations on microcontrollers and other hardware platforms without investing in bulky laboratory equipment. In addition, this approach also enables real-time inference from direct power measurement and on-device machine learning inference that, in the future, will lead to automatic cryptographic validation tools used in secure hardware design and certification processes.

Issues and limitations

Although the results show promising model performance and potential for real-time implementation several limitations have to be stated. Firstly the present study is restricted to DES encryption which is a comparatively straightforward and outdated encryption algorithm. Although adequate for controlled testing more recent encryption algorithms such as AES or post quantum algorithms need to be analyzed to confirm the applicability of the present approach.

Second, the models training data was gathered from uniform and controlled hardware runs. While real-time prediction was demonstrated how well the model withstands real world noise, voltage variations, or changing temperatures must be exhaustively tested. Hardware variation also poses issues: the power usage profile of the same algorithm can vary across microcontrollers or boards, impacting generalizability.

Another limitation is the models dependence on feature extraction. While statistical features (mean, max, min, std) of power traces resulted in high accuracy, the method may be losing valuable temporal leakage information that would be present with raw trace analysis or time-series models of higher complexity.

Ethical concerns and responsible use of side-channel tools

Though the study is aimed at creating a learning and security testing tool, the ethical aspect of publishing and releasing such methods should also be taken into consideration. Tools that can predict cryptographic keys from physical side channel information would be usable by the wrong people if they fell into the wrong hands. Their release and deployment must therefore meet high ethical requirements.

It is also critical to practice responsible disclosure in practical contexts. If the methods created here are extended to penetration testing or vulnerability analysis in actual devices legal and ethical standards need to guide how discoveries are revealed and remedied. Researchers also need to be forthcoming about the attack potential of such systems and not make it easier for exploitation against insecure or legacy systems still in active use.

Future directions in AI-powered side-channel research

This paper also points to some promising directions for future research in machine learning-based side channel analysis. One is the generalization of this approach to more sophisticated encryption algorithms, such as AES, RSA, and future post-quantum cryptographic primitives. These primitives imply greater computational complexity and thus might need more powerful feature extraction techniques or deeper neural network structures (e.g., CNNs or RNNs) to capture side-channel leakage patterns.

A further direction for growth lies in creating adaptive side-channel analysis frameworks. These might adapt on the fly to new devices, sensors, and environmental situations, rendering the approach still more generalizable and portable. Combination with unsupervised learning and anomaly detection could also allow systems to identify out-of-the-ordinary patterns without a priori labeling useful for embedded security monitoring. Further such systems can be taken one step further and transformed into active vulnerability scanners, which can be utilized at the design stage of embedded systems to model attacks and verify the effectiveness of countermeasures. Integration of such features in hardware testing workflows can render secure-by-design hardware development more practical and attainable.

Additionally, integrating such a pipeline with wireless communication and real-time monitoring (e.g., over Wi-Fi or BLE) can enable developers to receive constant feedback at firmware runtime, enabling secure embedded firmware development.

SUMMARY OF STUDENT CONTRIBUTION

My part in this research project was the design, development, and completion of an entire hardware-software pipeline for power-based side-channel attack (SCA) analysis of the DES encryption algorithm from start to end. My research work mainly involved the creation of a working and reproducible setup that could capture real-time power consumption during cryptographic operations and deduce sensitive information like plaintext and encryption keys using machine learning.

A major portion of my job was the hardware implementation, where I developed and designed the embedded system with microcontrollers like Arduino Nano and then ESP32. I was able to integrate the INA219 sensor module for measuring power, which allowed us to measure real-time voltage, current, power, and energy consumption while performing DES encryption. I also established the communication protocol via the Serial Bluetooth Terminal application, taking advantage of the ESP32's on-board Bluetooth functionality, for remote control of data logging via STARTLOG, STOPLOG, and GETLOG commands. In addition, I established SD card data logging and a modular framework for feature extraction from logged power traces.

On the software front, I dealt with the preprocessing and feature engineering phase where I implemented the extraction of relevant statistical features (mean, max, min, standard deviation) from the side-channel data acquired. These features constituted the input to the machine learning model.

Finally, I designed, trained, and tested an Multilayer Perceptron (MLP) model with TensorFlow and scikit-learn for plaintext and encryption key prediction in DES. I trained the model over 1000 epochs to get high accuracy rates 84% showing the promise of ML for power consumption leakage modeling. I also addressed model serialization and deployment problems of model reloading and data preprocessing with joblib.

CONCLUSION

The project aimed to investigate the viability and performance of power-based side channel attack (SCA) techniques in uncovering secret cryptographic data in this case from DES encryption based on real power consumption and machine learning. Adopting a multidisciplinary integration of embedded hardware, cryptography calculation, power monitoring through sensor-enabled techniques, as well as artificial intelligence, the project managed to illustrate a viable, low cost side channel analysis system that is capable of achieving high accuracy in predicting encryption keys and plaintext.

The project started with the addition of DES encryption and a microcontroller-based power and current monitoring circuit using the Arduino Nano and subsequently the ESP32. The transition to the ESP32 platform facilitated performance optimization, memory management, and wireless communication that supported real-time monitoring, Bluetooth triggered data logging, and SD card storage. Hardware setup employed the INA219 sensor to accurately log power consumption in encryption operations, while STARTLOG, STOPLOG, and GETLOG commands facilitated remote management of data logging by the user via the Serial Bluetooth Terminal app.

Latet meaningful statistical features like mean, maximum, minimum, and standard deviation of voltage, current, power, and energy were also extracted and arranged in a systematic dataset. That dataset was used as an input to train a supervised learning algorithm. A Multilayer Perceptron (MLP) was chosen for its capacity to discover non linear relationships in tabular data and was trained to forecast the plaintext as well as the encryption key values. The model worked well, with 84% accuracy in plaintext prediction at 1000 epochs representing how well the cryptographic operations correlate with power leakage.

To validate the feasibility of applying the trained model to real embedded environments, the model was deployed on a portable rig, where it performed live predictions on new power trace inputs, with the results being written to a 16x2 LED display. This not only served as proof of real world usefulness of the system but also showed the end to end feasibility of machine learning-based SCAs run on embedded hardware without needing high end lab equipment.

This work addresses a key gap in the literature by providing a modular low cost and reproducible platform for power based side channel attack research with particular focus on DES and embedded microcontroller platforms. It is both a research platform for cryptographic vulnerability investigation and an educational platform for instructing the threat of physical leakage in cryptographic implementations.

The success of this research opens up many lines of future research. These involve extending the model to cater to more sophisticated and ubiquitous encryption methods like AES evaluating the system against diverse environmental settings and noisy traces investigating more intricate neural network architectures like CNNs or LSTMs for processing time series data and designing countermeasures to mitigate such attacks. In addition to that the feature of real time anomaly detection, auto key extraction routines, and multi encryption mode support would further enhance the system's applicability to cryptographic hardware testing.

REFERENCES

[1] J. Hertz, "Understanding Side Channel Attack Basics," All About Circuit, 27 March 2022. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/understanding-.

[2] G. S. P. S. V. D. G.M, "An Overview of Acoustic Side-Channel Attack," 2020.

[3] Anysillicon , "How Differential Power Analysis (DPA) and Simple Power Analysis (SPA)," Anysillicon , [Online]. Available: https://anysilicon.com/semipedia/side-channel-attack-protection-against-differential-power-analysis/.

[4] S. M. Sim, "Differential Power Analysis on (Non-)Linear Feedback Shift Registers," 30 March 2020. [Online]. Available: https://eprint.iacr.org/2020/349.

[5] P. K. Joshua Ja e, "Differential Power Analysis," 1999. [Online]. Available: https://paulkocher.com/doc/DifferentialPowerAnalysis.pdf.

[6] R. H. Sloan, "Investigations of Power Analysis Attacks on Smartcards".

[7] E. O. Stefan Mangard, Power Analysis Attacks, 2007.

[8] C. O'Flynn, "ChipWhisperer An Open-Source Platform for Hardware Embedded Security Research," 2014. [Online]. Available: https://eprint.iacr.org/2014/204.pdf.

[9] F. X. M. T. Standaert, "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks," 2009. [Online]. Available: https://eprint.iacr.org/2006/139.pdf.

[10] L. B. Gabriel Zaid, "Methodology for Efficient CNN Architectures in Profiling Attacks," 25 June 2020. [Online]. Available: https://eprint.iacr.org/2019/803.

[11] S. B. Prasanna Ravi, "Machine Learning based Blind Side-Channel Attacks on PQC-based KEMs - A Case Study of Kyber KEM," 05 February 2024. [Online]. Available: https://eprint.iacr.org/2024/169.

[12] Y. G. Y. L. Q. Z. &. Y. Z. Huaxin Wang, "In-depth Correlation Power Analysis Attacks on a Hardware Implementation of CRYSTALS-Dilithium," 03 June 2024. [Online]. Available: https://cybersecurity.springeropen.com/articles/10.1186/s42400-024-00209-9.

[13] TECHINFO, "Navigating Vulnerability Disclosure: Best Practices from ISO/IEC 29147 and NIST SP 800-216," 03 November 2024. [Online]. Available: https://banglatechinfo.com/navigating-vulnerability-disclosure-best-practices-from-iso-iec-29147-and-nist-sp-800-216/.

[14] J. J. &. B. J. Paul Kocher, "Differential Power Analysis," *Advances in Cryptology — CRYPTO' 99,* vol. 1666, p. 388–397, 1999.

[15] B. G. E. D. M. I. V. &. J. V. Gabriel Hospodar, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering,* vol. 1, pp. 293-302, 2011.

[16] E. P. R. S. E. C. &. C. D. Ryad Benadjila, "Deep learning for side-channel analysis and introduction to ASCAD database," *Journal of Cryptoggraphic Engineering ,* vol. 10, pp. 163-188, 2019.

[17] Y. B. &. G. H. Yann LeCun, "Deep learning," 27 May 2015. [Online]. Available: https://www.nature.com/articles/nature14539.

[18] J. B. Diederik P. Kingma, "Adam: A Method for Stochastic Optimization," 22 December 2014. [Online]. Available: https://arxiv.org/abs/1412.6980.

[19] L. Lerman, G. Bontempi and O. Markowitch, "Power analysis attack: an approach based on machine learning," 01 January 2014. [Online]. Available: https://www.deepdyve.com/lp/inderscience-publishers/power-analysis-attack-an-approach-based-on-machine-learning-jJ6iUTGyoY.