FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Power side channel attack with a controllable power supply
Student:	Adam Verner
Supervisor:	Ing. Jiří Buček, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Security and Information technology
Department:	Department of Computer Systems
Validity:	until the end of summer semester 2023/2024

Instructions

Study the differential power analysis (DPA) attack.
Create a library or extend an existing one to control the Aim-TTi QL355TP lab DC power supply.
Use an oscilloscope and the controllable power supply to implement a DPA attack on a suitable microcontroller. The specific microcontroller will be selected after consultation with the supervisor.
Analyze the attack success rate depending on the voltage used and evaluate the results.
For your code, use Python with pyvisa and other suitable libraries.

Electronically approved by prof. Ing. Pavel Tvrdík, CSc. on 20 February 2023 in Prague.

Bachelor's thesis

POWER SIDE CHANNEL ATTACK WITH A CONTROLLABLE POWER SUPPLY

Adam Verner

Faculty of Information Technology
Department of Information Security
Supervisor: Ing. Jiří Buček, Ph.D.
May 11, 2023

Citation of this thesis: Verner Adam. *Power side channel attack with a controllable power supply.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Contents

List of Figures

List of Tables

List of code listings

I would like to express my deepest gratitude to my work supervisorr Ing. Jiří Buček for his patience and helpful insights during the course of the work. I am also grateful to family and friends for their support and motivation.

Declaration

In Prague on May 11, 2023 ....

Abstract

The thesis applies Correlation Power Analysis (CPA) attack against AES implementation running on a dedicated microcontroller and explores how the success depends on the operating voltage of the microcontroller. Part of the work deals with instrumentation required to execute the CPA attack on the target device. Data acquisition framework QCoDeS was extended to support the Aim-TTi QL355TP power supply. The CPA attack was applied on an Atmega microcontroller and proved to be successful although no direct correlation between operating voltage and attack success was found. Future work exploring the attack success against other types of crypto hardware was proposed.

Keywords Hardware security, Side channel attack, Controllable power supply, Correlation power analysis (CPA), Differential power analysis (DPA), Partial Guessing Entropy (PGE), QCoDeS framework

Abstrakt

Bakalářská práce aplikuje korelační odběrovou analýzu (CPA) na implementaci AES běžící na dedikovanéam mikrokontroléru a zkoumá závislost úspěchu na napětí mikrokontroléru. Část práce se zabývá ovládáním přístrojů potřebných k úspešnému provedení útoku CPA na cílové zařízení. Framework pro sběr dat QCoDeS byl rozšířen o podporu laboratorního zdroje Aim-TTi QL355TP. Útok CPA byl úspešně proveden na mikrokontrolér Atmega, ačkoliv přímá závislost mezi operačním napětím a úspěšností útoku nebyla nalezena. Byl nastíněn možný budoucí přístup zkoumající úspěšnost útoku proti jiným typům kryptografického hardwaru.

Klíčová slova Hardware bezpečnost, Útok postraním kanálem, Laboratorní zdroj, Korelační odběrová analýza, Diferenciální odběrová analýza, PGE, knihovna QCoDeS

List of abbreviations

SPA	Simple Power Analysis
DPA	Differential Power Analysis
CPA	Correlation Power Analysis
PSU	Power Supply Unit
PGE	Partial Guessing Entropy
DES	Data Encryption Standard
AES	Advanced Encryption Standard
DC	Direct Current

Introduction

Over the years, Cryptography has became omnipresent part of our everyday lives. Daily activities such as reading your email, posting on social media or even opening doors to you house using a smartcard are all made secure from malicious third parties thanks to various cryptographic protocols. Historically, substantial effort has been made to ensure the security of cryptographic algorithms, but even the most secure algorithms, which would take thousands of years to be broken, can be defeated in just a few minutes by using a side-channel attack. Most of the algorithms used today, such as Advanced Encryption Standards (AES), were not designed with the hardware implementation in mind and as such are usually susceptible to side-channel attacks.

The aim of this thesis is to study the Differential Power Analysis (DPA) side-channel attack. Using provided oscilloscope and controllable power supply the attack is mounted against a target microcontroller. Necessary scripts to control the provided instruments as well as the DPA itself will be implemented using Python. With the use of controllable DC supply as power source for the microcontroller, dependence of the success rate on the supplied voltage will be measured and analyzed.

Chapter 1
Preliminaries

This chapter briefly introduces the topic of the work. The First section id devoted to explanation of AES cipher and its operation. Following section explains the concept of Differential Power Analysis and steps necessary to mount such an attack. The third section mentions signal processing techniques relevant to DPA. And the last chapter shows the QCoDeS framework used for the data acquisition.

Most daily used devices which implement any sort of cryptography are made out of semiconductor logic gates, which are usually constructed using transistors. By applying or removing charge from a transistor's gate, electrons flow across the silicon substrate, resulting in the consumption of power and the emission of electromagnetic radiation. The resulting consumption of an integrated device corresponds to the activity it is currently performing as well as the data that is being used. For example on a processor with long data bus that uses pull up resistors the power consumption is significantly different when transferring $0xD8$ than $0x10$ (this is called Hamming Weight Model, other models are described later). While the different power consumption might not be directly interpretable (especially on more complex and noisy devices) the leaked information can accumulate over time exposing secrets. [1]

By closely measuring and analyzing the power consumed by a device during its operation, an attacker can extract secret information such as encryption keys. The first paper to explicitly discuss side-channel attacks on cryptographic systems using power analysis (and not just timing) was published by Paul Kocher and Joshua Jaffe in 1999 [2], in which they described a technique for extracting secret keys from smart cards by measuring the power consumption of the card during encryption operations.

First, the Advanced Encryption Standard (AES) is introduced, followed by a brief explanation of the DPA method in section 1.1. Furthermore, an introduction to the QCoDeS library, which is later used to control the measurement equipment, is given in 1.5.

1.1 Differential Power Analysis

The first step when performing Differential Power Analysis (DPA) is to collect multiple *traces* from the targeted device. A trace is a sequence of power consumption measurements made while the device is performing cryptographic operations.

DPA is a statistical method used to identify data-depended correlation between different traces. An assumption is made about the data first which is used to divide the collected traces into two groups. If the assumption about the data is correct (correlates with the measurement contained in the traces) the square mean difference between each group will be equal to non zero value (a distinct peak in DPA trace). On the other hand if the assumption is incorrect and does not correlate with divided traces the square mean difference will approach zero as the number of traces increases. Additionally with the increased number of traces available the noise present in the measurement will be canceled out allowing tiny correlations to be isolated. [2]

The crucial step in DPA is to choose the correct selection function. The function is used to divide the traces into two distinct groups, and averages of each group are then used to calculate the DPA trace. If the DPA trace shows significant spikes, the selection function was chosen correctly and the value was actually computed by the target device. If no significant correlation is observed in the DPA trace, either not enough traces were gathered in order to observe the small correlation or the selection function was chosen incorrectly. [1]

Depending on the architecture of the device, the selection function may either be hypothetical value of an output from the combinational logic or from more complex operation such as state change in a register. The selection function used by DPA must be a binary function as the data needs to be divided into two subgroups.

1.1.1 Correlation Power Analysis

Correlation Power Analysis (CPA) is an extension to DPA which works by evaluating the degree of correlation between variations within the set of measurements. [3] when performing CPA it is crucial to produce accurate power model of the targeted device in order to succeed.

There are two consumption models typically used in CPA applications:

Hamming Weight works by mapping the state value to the number of non-zero bits. For example intermediate value `0xd5` has Hamming Weight equal to 5.

Hamming distance is a measure of the number of bits that differ in value between two state changes. For example if an intermediate vale $0x3f$ changes value into $0x76$ the Hamming distance is equal to 3.

The CPA attack is done by constructing a hypothesis for each possible value of the state and then searching for correlation across the set of all measured traces. In practice the CPA against AES with the knowledge of plaintexts used for encryption can be implemented as follows. [4]

First a matrix t is constructed, where each row represents single power trace measurement, giving us a matrix with total dimensions of $N \times L$ where N is the number of traces available and L is the number of samples in each power trace. For simplicity first byte of encryption key k_0 is used. Secondly matrix h is constructed with hypothesis about the inner values of the cipher for each trace using the Hamming Weight power model. The dimensions of h are $N \times D$ where N is the number of traces and D is number of possible values the k_0 can have. Each row in the matrix h corresponds to a single trace measurement and the rows repent different hypothesis about the key value such that $h_{i,j} = HW(S(d_i \oplus j))$ where S is an unary function representing the AES *SubBytes* operation, HW is a function mapping the intermediate values to the amount of non-zero bits and $d_{i,0}$ is the value of the first byte in the plaintext corresponding to ith trace. Both matrices are visualized in the figure 1.1 for better understanding. [4, 5]

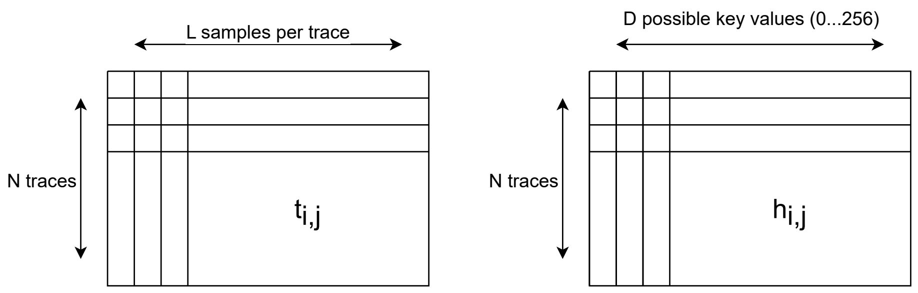


Figure 1.1 Representation of the matrices with measured data (left) and key hypothesis.

To find the correlation between the hypothetical values and measured traces, Pearson's correlation coefficient is used. [6] When applied to the matrices as sample coefficient it can be obtained using the following formula:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

The columns in the resulting matrix correspond to a sample at time i and each column corresponds to a different key hypothesis. The correct key can then be found simply by searching for the cell with the highest value and the row will correspond to the key value used for the hypothesis.

1.2 Partial Guessing Entropy

Partial Guessing Entropy (PGE) is a useful metric used to measure the information leakage in cryptographic devices, such as microcontrollers, during DPA attacks. The correct key has to be known in order to evaluate PGE. Guessing entropy for a specific byte of the key is obtained by sorting all the possible key values by their probability and taking the index of the correct key, 0 indicates that the keys is known. All calculated entropy values are then used to calculated PGE of an attack by finding the average. [7]

1.3 Advanced Encryption Standard

AES is block cipher also known by its original name Rijndael, developed by two Belgian cryptographesrs, J. Daemen and V. Rijmen. [8] A variant of the cipher has been approved by NSA and included in the ISO standard 18033-3. [9] The cipher has a fixed block size of 128 bits, three different configurations can be used with 128 bit key and 10 rounds, 196 bit key and 12 rounds, and 256 bit key and 14 rounds.

The encryption algorithm can be divided into four distinct steps:

Key Expansion Each round of the cipher requires 128 bit key (equal to the block size) which are derived from the initial cipher key.

Initial round key addition the first derived key is XORed with the initial state.

Encryption Rounds each encryption round consists of four sub-steps:

- *Sub Bytes* – each byte is replaced using lookup table,
- *Shift Rows* – each row is shifted with constant number of steps,
- *Mix Columns* – each column is lineary transposed using matrix multiplication with static value, and
- *Add Round Key* – state is XORed with corresponding derived key (same as step 2).

Final round The last round is same as previous rounds except the *Mix Columns* operation is excluded.

When attacking AES the SubBytes operation is targeted as it is a nonlinear operations. The non-linearity leads to observable power variations that leak information about the secret key. In the figure 1.2 the two important parts of AES which can be targeted by DPA are displayed. Based on the knowlege of either plaintext or cipher text the beginning of AES operation or the end is choosen. The state value which is targeted in either scenario is denoted by h, the values of the block key are denoted by k. [1]

1.4 Signal processing

Differential Power Analysis requires the processing of a relatively large amount of data. In order to extract meaningful information from the measured traces, some form of signal processing has to be performed. three steps are mentioned, that all may be done in order to mount a successful attack – noise reduction, trace alignment and data reduction. These processing techniques are critical in DPA attacks as they help to improve signal-to-noise

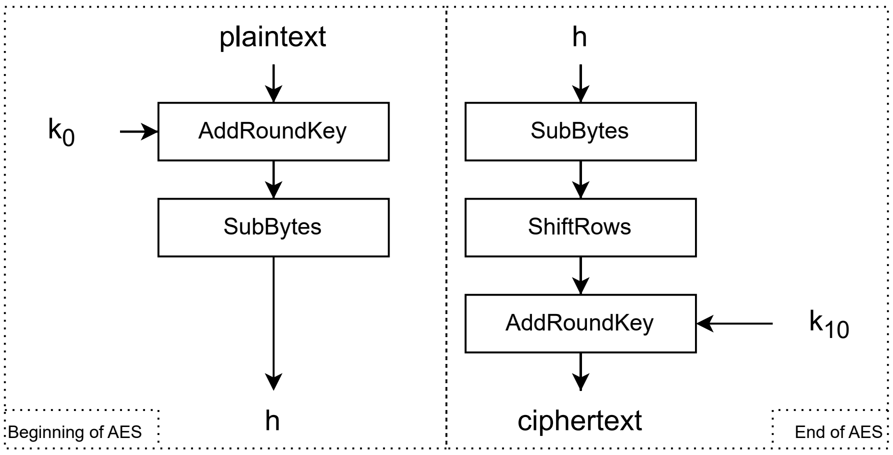


Figure 1.2 Internal values of AES which can be target by DPA attack.

ratio, remove unwanted noise and make the amount of computation feasible. The choice of which options to choose depends on the application, knowledge and various factors of the system, usually there is not a single *correct way* and more options have to be verified experimentally.

1.4.1 Noise reduction

The first step when it comes to signal processing in general is to remove unwanted noise from the captured signal traces. The noise in the signal can arise from various sources such as measurement equipment, electrical interference or other environmental factors. Removing noise from the signal can improve accuracy and decrease the number of traces required for successful attack. There are several denoising techniques that can be used, most notably:

- Low-pass or band-pass filtering
- Median filtering
- Order filtering
- FFT or wavelet denoising

Each method has its strengths and drawbacks, however, providing their detailed explanation is out of the scope of this work. Implementation of all the aforementioned denoising functions are available in the Python library *SciPy*. [10]

1.4.2 Trace alignment

When performing DPA, trace alignment is crucial because it ensures that the power traces are properly aligned with the corresponding parts of the cryptographic algorithm being executed on the target device. If the traces are not properly aligned, then the DPA will not succeed or it will identify non-existent correlations and cause false-positives.

In all cases the first trace is selected as an anchor and then every other trace is shifted in order to match the anchor. Naive approach consists of finding a global maxima in both traces and them using the location of both to calculate the shift necessary to match the traces. This method is extremely time and space efficient, however, rarely functional. Slightly more complex technique, is to calculate cross-correlation of the two traces, the shift between the two traces is equal to the point where the value of the cross-correlation is the highest. [11] This method is called *Cross-correlation alignment* and works well for traces that are misaligned only by shift in time, usually because of imprecise or missing trigger signal, it's use is demonstrated in figure 1.3. More advanced alignment techniques include *Dynamic time warping* or *Elastic alignment*[12] – strategies often used in speech recognition – to match skewed traces usually produced by some DPA countermeasures or unstable clocking signals.

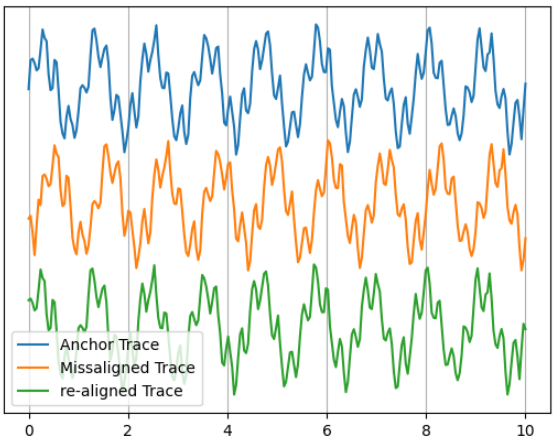


Figure 1.3 Illustrative example of trace alignment done using cross-correlation alignment

1.4.3 Data reduction

The third step is to reduce the number of data that will be processed in order to make the amount of computation feasible. It is important especially when working with large number of traces and/or traces containing lots of sample points.

Windowing method which is pretty intuitive, but requires some information about the execution, selects the part of the trace which does relevant computation to the attack. For example when attacking AES implementation the first round could be identified and kept without the rest of the computation.

With *down sampling* the amount of data can be reduced as well, the downside of it is that some information from the signal is lost and with it a part of the leaked information as well.

Provided information about the hardware design of the target system is available, more informed method of downsampling can be done. In the case of ATMega8, the operands that are used in the registers are always fetched within the same offset from the rising edge of the source clock, as can be seen in the figure 1.4. The rest of the clock cycle should be irrelevant, thus only the part of the recorded trace that is moving the secrets across the data bus and causing information leak[1]. This reduction method is similar to the process of template-based attacks. [13]

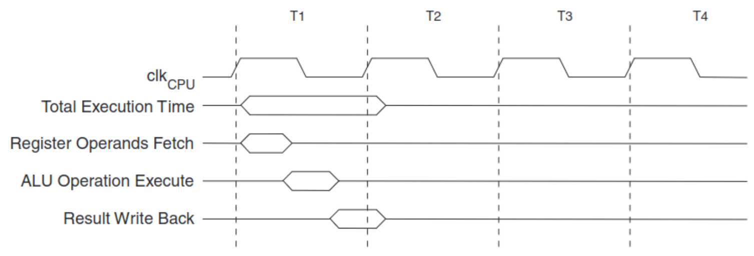


Figure 1.4 Internal timing concept for the Register File (taken from [14])

1.5 QCoDes library

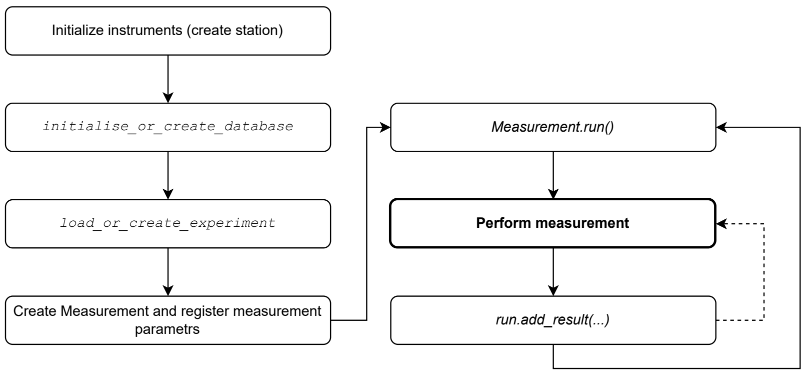
QCoDeS is a Python-based data acquisition framework developed by the Copenhagen / Delft / Sydney / Microsoft quantum computing consortium. While it has been developed to serve the needs of nanoelectronic device experiments, it is not inherently limited to such experiments, and can be used anywhere a system with many degrees of freedom is controllable by computer. [15]

The QCoDeS framework is a great option to control the measurement workflow of the experiment. It provides easy access to data persistence using local SQLite database. The whole framework is well documented, including a large number of usage examples for all aspects of the framework ranging from simple measurement up to custom driver development. Pyvisa [16] library is used in the background to communicate with the measurement instruments, the drivers usually expose the handler in the `.visa_handle` parameter. The drivers implemented by the library can be used as standalone components without the rest of the framework.

The framework which uses PyVISA in the background was preferred to PyVISA as it offers better management of complex instrument control, such as that of oscilloscopes and power supplies. It hides the SCPI commands behind a layer of abstraction without losing direct access to the communication library. The ecosystem provides data retention capabilities useful for long-running experiments.

In the figure 1.5 typical workflow using QCoDeS is described. The usual steps when performing measurement using the framework are explained below.

[1]Experimentally verified during the implementation


Initialize instruments (create station)
initialise_or_create_database
load_or_create_experiment
Create Measurement and register measurement parametrs
Measurement.run()
Perform measurement
run.add_result(...)

■ **Figure 1.5** Typical QCoDeS data acquisition workflow

The first step consists of instrument initialization. In our case this means connecting to the Oscilloscope and Power Supply with their designated drivers. The details of the communication protocol and connection options are described later in the document for both measurement devices – Power Supply class is described in 2.2.1 and Keysight Oscilloscope connection support is described in 2.3.1.

Visa addresses are required for both instruments, these can be either Universal Serial Bus addresses (starting with prefix 'USB') e.g.: `USB0::0x2A8D::0x1766::MY60104433::0::INSTR` or network addresses e.g.: `TCPIP0::10.11.58.253::9221::SOCKET`. There are also other VISA resource names, but none of them are relevant as the are not available in the laboratory.

The second step is simple and only consists of database initialization. This step is handled entirely in the background. If the database does not exist it is created, if it exists, but the revision is from older QCoDeS version it will be automatically updated otherwise it is kept as is, containing all the previously made experiments. The database connection feature is really useful, because it stores data on disk instead of random access memory which is far more limited and also provides data persistence on power outage or script failure.

The third step is fairly similar to the previous one. The experiment is created (there can be multiple experiments in the same database) and each subsequent measurement is tagged with the experiment name so it can be easily searched in the database. The grouping of measurement runs into different experiments is offered as a convenience, but it is not essential to any functionality provided by QCoDeS and can be ignored or used with default values.

In the fourth step `Measurement` class instance is created and variables that will later be recorded are registered. There are two options on how to register a parameter into the measurement, either by using a parameter of an existing device driver (from the first step) e.g.: `supply.ch1.volt` or instance of a Parameter class can be used.

The fifth, sixth and seventh step are all part of the *measurement loop*. These three steps together are demonstrated in the example 1.1. The measurement loop is

Code listing 1.1 QCoDeS measurement loop example

```
with measurement.run() as datasaver:
    for x in range(5, 15):
        supply.ch1.set(x)
        volt = multimeter.v1.get()
        datasaver.add_result((supply.ch1, x),
                             (multimeter.v1, volt))
```

wrapped using the Pythons `with-context` statement and each measurement is then saved using the `add_result` method available from the context manager.

Code listing 1.2 Data retrieval from database previously created using QCoDes

```
>>> qc.initialise_or_create_database_at('Measurement.db')
>>> guids = qc.get_guids_by_run_spec(
        experiment_name='EX1',
        sample_name='sample1'
        )
>>> for guid in guids:
        dataset = qc.load_by_guid(guid)
        data = dataset.get_parameter_data()
        print(data['voltage'])
```

1.5.1 Measurement retrieval

The Library provides easy way to access the measurement results using provided helper functions. In the final measurement combination of functions `get_guids_by_run_spec` and `load_by_guid` is used to iterate through all the measurements that were done. The `get_guids_by_run_spec` function takes various filter criteria as parameters, experiment name for example and then returns all the measurement GUIDs that match this criteria. The dataset can then be loaded from the database into memory using the function `load_by_guid`. The code listing 1.2 demonstrates how to retrieve traces stored in the experiment database.

1.5.2 Driver development

Writing custom drivers for QCoDes is easy as the library provides high-level interface. Base class `VisaInstrument` is available, which handles the initialization of the communication with the instrument using the underlying pyvisa library. Parameters of the implemented instrument are added using the `add_parameter` function which is provided by the baseclass. In case it makes sense, instance of `InstrummentChannel`, which behaves identically as `VisaInstrument`, can be used to group parameters together. This usually makes sense in case of devices such as multi-channel power supplies. When publishing a driver to the original repository, it should be stored in a file with following name convention: `qcodes\instrument_drivers\[Vendor]\[Vendor]_\[Model].py`

Chapter 2

Measurement setup

This chapter is devoted to description of the measurement setup, individual devices in the system and overall connection. In first section target microcontroller is introduced, the features of the firmware are described and notes on how to program the microcontroller are given. The following section describes the features of the Aim-TTi QL355TP Power Supply, it's communication protocol and device driver support. The third section Introduces the oscilloscope used for trace measurement, it's mode of operation and communication.

Measurement setup consists of four parts, the Host computer, Power supply, Target device and Oscilloscope as can be seen in the setup overview in figure 2.1. The Host computer controlls the flow of the whole experiment, sets up measurement instruments and saves recorded data. Power supply generates voltage for the target device to run on as instructed by the Host. Randomly generated plaintexts and keys are sent into the target device for encryption. And the oscilloscope measures the devices power consumption.

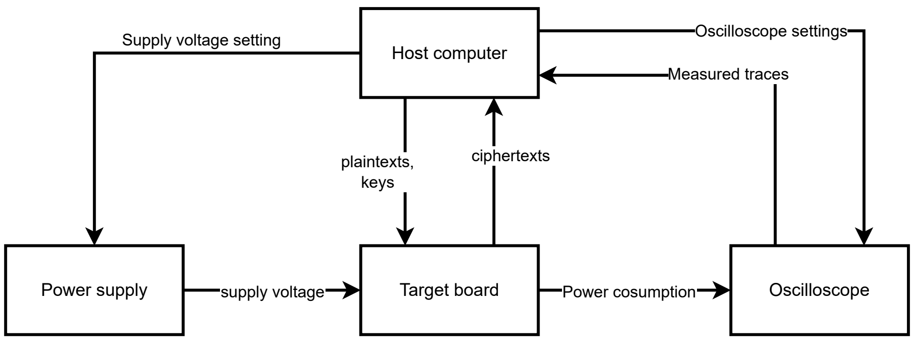


Figure 2.1 General system connection overview

2.1 Target device

As a target device 8-bit Atmel AVR Microcontroller *Atmega8* was selected. It features RISC architecture, has 8K bytes of on-chip flash memory which can be self-programmed and 1K byte Internal SRAM memory. The microcontroller can operate on voltages raging from 2.7 V up to 5.5 V, with disabled brownout detection the chip can operate on lower voltages but with no guarantee. Despite being fairly limited in terms of available memory and processing speed the ATmega8 is fairly popular microcontroller because of it's low price and high availability as well as minimal requirements on any additional on-board components.

In order to ensure proper and stable functionality, especially under near-limit conditions, the microcontroller is connected to other circuitry. Breadboard has been used to allow simple and modifiable setup. The used circuitry in the system includes mainly the following:

- External 8 Mhz crystal oscillator
- CP2102 USB to UART module
- LM358 comparator [17]
- Current measurement resistor

Brief description of reasons as to why each component was included in the system are given below.

External oscillator was used in order to ensure more stable clocking signal with minimal jitter than what is available from the on-board oscillator as stable clocking signal is essential for DPA.

USB to UART module provides a gateway for the necessary communication with the host computer – sending and receiving commands.

Comparator was used in order to boost the level of data transmission signal coming out from the device as its voltage was not sufficient for the CP2102 module when the microcontroller was operating on the lower limit of supply voltage.

current measurement resistor between GND and the microcontrollers own ground is necessary for current consumption measurement.

Schematic drawing of connections between each component required to operate the microcontroller as well as measurement equipment connections are displayed in figure 2.2. Final picture of the components connected on a breadboard can be seen in the figure 2.3.

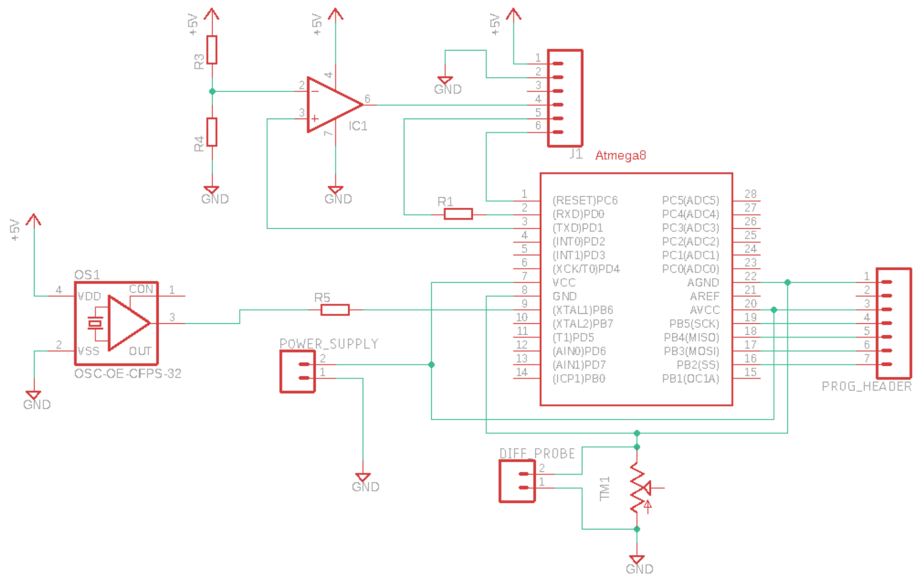


Figure 2.2 MCU connection schematic

2.1.1 Firmware

The microcontroller is first to be loaded with some form of bootloader so that it can later be programmed without using dedicated programmer (e.g.: T-LINK/V2). The project uses Bootloader described in the AVR109[18] application note from Atmel, which enables the device to be able to self-program (write both Flash and EEPROM Memories and set required lock bits). In the aforementioned document the communication protocol and all supported features are described. The implementation of the bootloader is available online[1] and can be compiled using Atmel studio 7 or equivalent tooling. The communication between the host computer which is used to control all the tooling and the bootloader is handled by a program called avrdude. [19]

In the final setup the bootloader was not used and direct programming using ST-LINK was employed instead, because of compatibility issue between the available bootloader code, chip revision and avrdude version. Since frequent re-programming of the board was not required – the sample program was successfully verified and no bugs were discovered – the microcontroller did not need to be reprogrammed at all.

2.1.2 AES implementation

Program containing AES implementation was provided by the supervisor [20] as is and without any provisions.

Copy of the provided source code is located in the file `mega8aes.zip`. The folder `GccApplication1` contains the necessary project files to be imported into

[1]https://www.microchip.com/en-us/application-notes/an1644

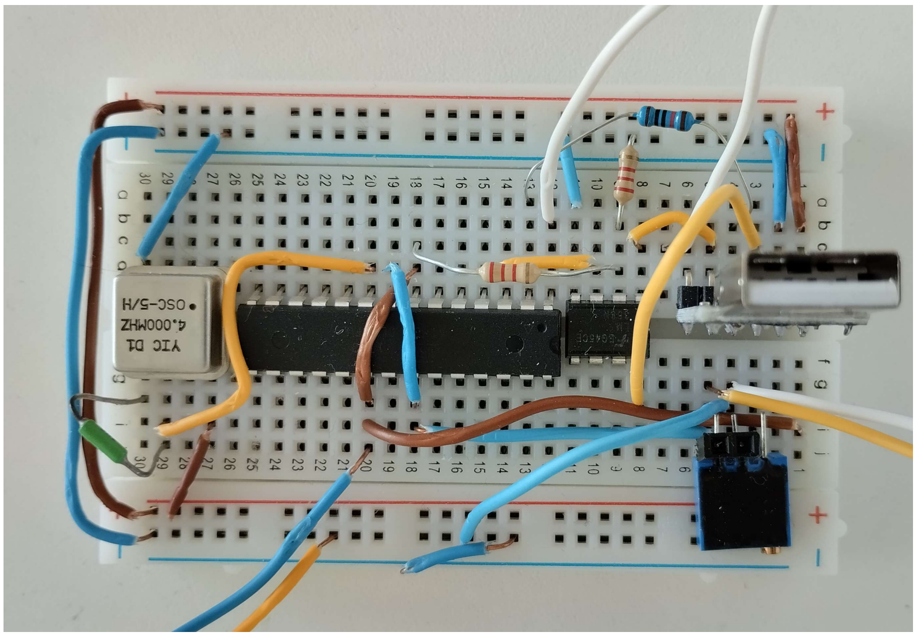


Figure 2.3 Image of the board setup during measurement with main components highlighted.

Microchip Studio. Files with the implementation of AES – `aes.c` and its header file `aes.h` – is located in the root of the folder. Main function of the application is located in the file `testdpa_uart.c`. Compiled binary as it was flashed on the target device is located in the output directory of the Microchip Studio at `GccApplication1GccApplication1Release`.

The AES implementation correctness has been first verified against the AES implementation provided by the Python library PyCryptodome. [21] Set of random plaintexts and keys were used to generate ciphertexts using the device as well as host computer and the outputs of both were verified. All proved to be the same and no errors were detected. Later on, during the measurement operation the encryption correctness was also verified to make sure the chip is operating correctly, which was important especially when working near the operational limits of the microcontroller.

Multiple commands are supported by the firmware all of which are shown in the table 2.1. Most important commands used in the work were “p”, “k”, “r” and “C” used precisely in that order. These command functions are fairly straight forward, the *receive* commands store the next incoming 16 bytes into the respective inner buffers, then the encryption command calls the AES implementation on the device which uses the previously stored values, calculates the ciphertext and stores it into another buffer, which is then read out by the host out using the transmit command.

Since the firmware generates pulses on PB1 precisely at the start of AES operation no trace alignment needed to be done as all the traces match closely in their timing (also thanks to stable power source there is no skew on the external oscillator). If that was not the case and the traces need some for of alignment, generally two types approaches are available. First approach works by shifting the traces in time according to distance

Command	action
“p”	receive plaintext
“p”	send stored plaintext
“c”	receive ciphertext
“C”	transmit ciphertext
“k”	receive encryption key
“K”	transmit encryption key
“r”	encrypt plaintext using stored key
“R”	generate random plaintext
“b”	jump into bootloader
default	print usage

Table 2.1 Microcontroller Firmware commands

between global maxima in the area of the traces which are targeted by the attack. The second approach is more complex and uses so-called *elastic alignment* – strategy often used in speech recognition – to match skewed traces usually produced by simple DPA countermeasures. [12]

2.2 Power Supply

General requirement on power supply that is used is to produce as little noise as possible in order not to influence the measured data. Manufacturer of the used power supply QLP355 promises Linear regulation with noise below 0.35mV rms. [22] which is well bellow the measurement capabilities of the used Digital Storage Oscilloscope used.

The model incorporates two single output units and one auxiliary low voltage output. Output units support $1mV$ setting resolution across the whole output range of 0-30 volts. The instrument can be controlled remotely via one of its many interfaces, these include RS232, USB, LAN and GPIB.

RS232 interface requires no setup at all and uses standard fully wired cable without any cross-over connections. Because the connection requires XON/XOFF handshake only ASCII encoded data can be send, but since all the parameters are sent in decimal format this is not an issue. The baud rate can be configured from 600 up to 19200, with **default baud rate 9600**, the value can be changed using the physical interface on the device, other parameters are described in the documentation and should be same as default values in most standard implementations.

USB interface requires a Communication Device Class driver, on computers running Windows, suitable driver is provided by Microsoft and should be installed automatically by Windows plug and play function when the device is connected to the computer for the first time. On Linux the driver for the USB controller is included in the kernel and works without any setup, node the device has been bound to can be seen in the system log when connection is made, typically as */dev/ttyUSB0*. The Baud rate and other settings are unnecessary and are ignored by the USB Controller.

Command	description
$V < N >< NRF >$	Set output $< N >$ to $< NRF >$ Volts.
$OVP < N >< NRF >$	Set output $< N >$ over voltage protection to $< NRF >$ V
$TRIPRST$	Attempt to clear all trip conditions from all outputs
$I < N >< NRF >$	Set output $< N >$ current limit to $< NRF >$ Amps
$OP < N >< NRF >$	Set output $< N >$ on/off
$*IDN?$	Returns the instrument identification

Figure 2.4 Partial list of commands supported by the AimTTi QL355TP power supply

LAN interface is LXI (Lan eXtensions for Instrumentation)[23] compliant and remote control of the interface is possible using TCP/IP protocol. The instrument provides a basic Web server which can be used to read device information, configuration as well as an option to directly send commands. Settings of the interface can be switched back into factory mode using switch on the rear panel of the unit. When connected the device uses DHCP to obtain its IP address, in the unlikely scenario that DHCP is not available the devices uses static IP address of **192.168.0.100**. ICMP Ping and VXI-11 Discover Discovery protocols are implemented for instrument discovery, tools such as Keysight Command Expert can be used to locate the device on local network.

GPIB interface is also provided and well described in the documentation, but since it has no relevance to this work as it could not be used in the laboratory it's description is omitted.

All of the interfaces implement Standard Commands for Programmable Instruments (SCPI). The device executes the commands in order as they are received and responses to query commands are sent immediately. List with detailed description of over forty commands is given in the documentation ranging from output setting to interface locking. Some commands that can be used are listed in the 2.4, for example command to set Voltage on the first output to 5.324 volts would be "$V15.324\n$".

2.2.1 QCoDeS support

The QCoDeS library did not offer the support for QL355TP out of the box, however the documentation explains in great detail how to write custom drivers for new devices that communicate using VISA protocol. In the folder with instrument drivers a number of similar devices manufactured by AimTTi has been found. Most of the instrument drivers derive from the class *qcodes.instrument_drivers.AimTTi.AimTTi* which provides interface for most of the SCPI system commands as well as initialization for channel parameters. Unfortunately the class did not support QL355TP as it was not aware of the number of channels the device has neither did it implement any form of auto-detection.

Support for QL355TP was added into the library, which required some minor changes to the `AimTTi` class, which other drivers for Aim-TTI power supplies derive from. Entry for the power supply had to be added into the parent class as it requires information about the number of channels the device has. This was previousely not done in easily-extensible way so the class was modified. These changes has since then been accepted

Code listing 2.1 Patched AimTTi class usage demonstration

```
from qcodes.instrument_drivers.AimTTi import AimTTi
AimTTi._numOutputChannels['QL355P'] = 2
pws = SupplyDriver(
    'power␣supply', 'TCPIP0::10.11.58.253::9221::SOCKET'
)
Connected to: THURLBY THANDAR QL355P (serial:552321) in 0.03s
```

by the maintainers of the library and merged into the master branch of the project[2], they were included in the latest (as of writing) release version 0.38.1 and are available in Python Package Index PyPI.

As a part of the patch, the parameter_numOutputChannels was exposed allowing potential users to extend the lookup table with information regarding other devices, which are not supported by default. In the code 2.1 the aforementioned feature is demonstrated with the usage of `QL355P` device which, unlike `QL355P`, is not explicitly supported by QCoDeS.

Additionally support for over current protection, over voltage protection, and trip reset was added in to the library. The implementation of these features required:

- adding new parameters into the instance of a class implementing communication with the channels,
- create custom parser for the response returned by the over voltage and current protection query.
- add custom function which triggers the device trip reset.

The implementation of these features is provided in the file `AimTTiPatched.py`. Request to the maintainers of QCoDeS to merge the changes into the upstream has been made, but as of writing it has not been approved yet. Although, that should not be a problem once someone from the maintaining team is assigned the pull request

▶ Note 2.1. `https://github.com/QCoDeS/Qcodes/pull/5156`.

[2]`https://github.com/QCoDeS/Qcodes/pull/5021`

2.3 Oscilloscope

Keysight DSOX3024T [24] from InfiniiVision family was used. It offers sampling speed of upto 200 MHz which is sufficient for the application as the MCU is running on 8 Mhz clock signal. Remote control is available over USB providing a way to programmatically change running configuration allowing a completely automatized setup for various ranges of measurements.

The current consumption of the microcontroller changes with the supply voltage according to the figure 2.5, although the manufacturer does not test these values during production. This imposes an inequality between the measured data for various supply voltages. One option is to compensate the signal values later in software using some transformation method e.g.: linear transformation, but this would introduce more noise into the data and some information could be lost. Other, much better option is to compensate the signal levels using the analog amplifier that is in the oscilloscope which is better suited for this application. The oscilloscope settings that which were acquired manually using the on-screen measurement options (peak-to-peak measurement, averaging, ...) can be found in the table 2.6. The voltage and time range are for the whole oscilloscope (reasons are mentioned later in the chapter) not per one square grid. For more complex measurement setups this step could be easily automated, but for the amount of settings that were required in this application it was not necessary.

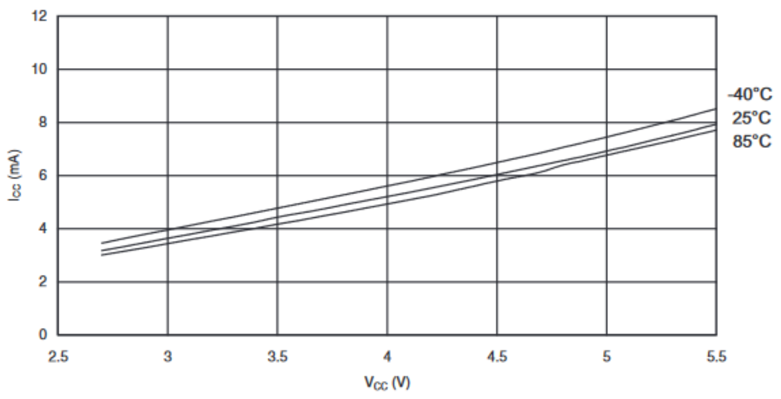


Figure 2.5 Supply Current vs. V_{CC} (taken from datasheet [14])

2.3.1 QCoDeS support

There is no out-of-the box support for any oscilloscope from the *Keysight InfiniiVision* family in QCoDeS, however, there is support for higher series of oscilloscopes called *Infiniium*. The protocols used by both product families are similar to each other and overlap in many important features. The file `KeysightInfiniiumPatched.py` contains the patched driver, which can be used to controll some of the features provided by the oscilloscope. When initializing the driver and connecting to the device multiple warning are shown in the output, but they can be safely ignored.

Supply voltage	Horizontal Range	H. Offset	Vertical Range	V. Offset
2.25 V	180 mV	067.8 mV	280 μs	1.36 ms
2.50 V	160 mV	076.2 mV	280 μs	1.36 ms
2.75 V	180 mV	092.4 mV	280 μs	1.36 ms
3.00 V	180 mV	108.4 mV	280 μs	1.36 ms
3.25 V	190 mV	126.3 mV	280 μs	1.36 ms
3.50 V	210 mV	144.3 mV	280 μs	1.36 ms
3.75 V	220 mV	162.7 mV	280 μs	1.36 ms
4.00 V	220 mV	192.4 mV	280 μs	1.36 ms
4.25 V	220 mV	228.8 mV	280 μs	1.36 ms
4.50 V	220 mV	266.3 mV	280 μs	1.36 ms
4.75 V	220 mV	299.4 mV	280 μs	1.36 ms
5.00 V	220 mV	339.4 mV	280 μs	1.36 ms
5.25 V	220 mV	398.9 mV	280 μs	1.36 ms

Figure 2.6 Oscilloscope Channel 2 settings

Code listing 2.2 DSOX3024T oscilloscope capture setting and run

```
>>> osc = ScopeDriver('scope', ADDR_SCOPE, timeout=2)
Connected to: KEYSIGHT TECHNOLOGIES DSO-X 3024T (serial: [...])
>>> osc.ch1.range(0.460 * 8)
>>> osc.timebase_range(280e-6  * 10)
>>> osc.trigger_edge_source(f'CHAN{1}')
>>> osc.trigger_edge_slope('POS')
>>> osc.ch1.trigger_level(0.75)
>>> osc.ch1.display(True)
>>> osc.single()  # osc.run for Auto
```

Channel capture, trigger and timebase settings are same for both families allowing the use of the driver for those features without any additional work or patching. The protocol does not expose all of the available functionality of the oscilloscope, especially advanced trigger features and channel input processing. Its worth mentioning that the settings used in the driver are always meant for the whole screen unlike when they're displayed on the oscilloscope where they show as values for one displayed square on the display. In practice that means values that are displayed on the oscilloscope screen for the Y axis (Voltage range) must be multiplied by the number of squares on the screen, which is 8 for DSOX3024T and time setting on the display has to be multiplied by 10 when using the driver. Example setup of single capture run without is shown in code listing 2.2, where 460 mV and sampling time of 280 μs per square is set as well as negative edge trigger. trace reading is missing in the example as that will be discussed later.

Part of the Infiniium communication protocol which is responsible for data reading (captured trace retrieval) contains parameters which the lower end InfiniiVision does not offer in it's protocol and on which the drivers implementation relies making it unusable in that scenario. However, the VISA handler used for the communication is exposed by the driver allowing commands to be sent directly into the device and to read raw output. When reading raw bytes with the trace data the handle allows *chunk_size* to be specified which leads to efficient burst reading from the device. Each point in the

Code listing 2.3 Reading data directly into numpy array from channel 1 using the exposed VISA handle

```
>>> osc.write(":WAVeform:POINts␣MAX")
>>> osc.write(f":WAV:SOUR␣CHANnel{1}")
>>> osc.write(":WAVeform:FORMat␣BYTE")
>>> osc.write(":WAV:DATA?")  # ask for data
>>> digits = int(osc.visa_handle.read_bytes(2)[1:])
>>> length = int(osc.visa_handle.read_bytes(digits)) + 1
>>> trc = osc.visa_handle.read_bytes(length, chunk_size=length)
>>> array = np.frombuffer(trc[:-1], dtype="uint8")
```

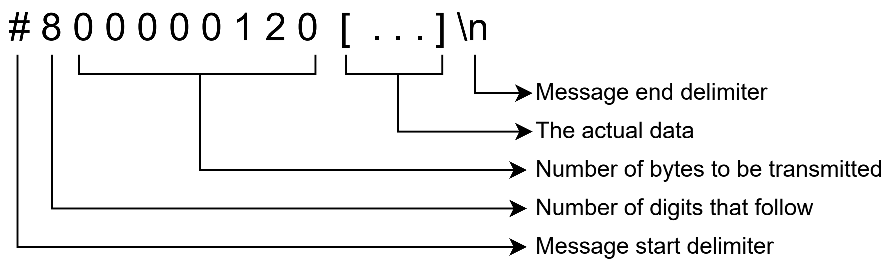


Figure 2.7 Format of the data returned by `:WAV:DATA?` command [25]

trace is represented by one byte since the oscilloscope has 8 bit precision ADC and *numpy.frombuffer* can be used to load the raw binary data into an array structure suitable for processing. Example of direct data reading is show in the code listing 2.3.

The format of the data returned by the oscilloscope is broken down in the figure 2.7, other relevant SCPI commands are described in the programmers guide released by Keysight. [25]

Chapter 3

Attacks on the target device

Systematic measurement encryption cycles has been performed on the target device with various supply voltage levels. DPA attack was launched successfully in every instance with different results. And the measured data has been thoroughly analyzed.

The whole measurement procedure with detailed description of each step is available in the Jupyter notebook *Measurement.ipynb* which has been used to control the whole experiment. The notebook consists of multiple parts, first the measurement instruments are initialized and communication with the microcontroller is verified using `chip_operation` function. Settings for the oscilloscope are acquired in a semi-automatic manner requiring some input from the operator. Afterwards the measurement loop is executed for each voltage level and the produced traces are stored. The rest of the notebook is made up by analysis of the measured data.

Some of the helper functions, which used in the experiment, are located in the file *utils.py*. Raw data as retrieved from the oscilloscope is stored in the sqlite database generated by QCoDeS in the file *Measurement.db*, the data can be accessed using the QCoDes library as described in the section 1.5.1.

Initially, the data needed for the experiment was recorded in the laboratory producing 14 sets of data across different voltages ranging from 2.25 V up to 5.5 V. Before recording each set, non-measured cycles of encryption were executed on the device in order to stabilize the power consumption and charge in all present capacitors to a stable level. Due to the $\approx 9\ MB$ size of each trace as received from the oscilloscopes, and the need to record large amount of traces, the length of each trace had to be restricted to conserve storage space used by the database. The part of the trace which was kept was chosen in a way to include the first round round of the AES encryption, which is the target of the DPA implementation. Without this restriction, the size of the database would have exceeded tens of gigabytes making it impractical to store and analyze.

As a preliminary step, DPA was performed on the raw traces for each set and successful attacks were observed in every case. Minimal number of traces required for a successful key recovery was calculated for each set by reducing the amount of traces used in the DPA and observing the success of the attack, additionally binary search was used in order to reduce the amount of computation needed from $\mathcal{O}(n)$ where n is the number of traces in each set to $\mathcal{O}(\log n)$. The outcome can be seen in the figure 3.1, which does not exhibit any clear relationship between the supply voltage and the minimal number of traces required for successfully DPA attack.

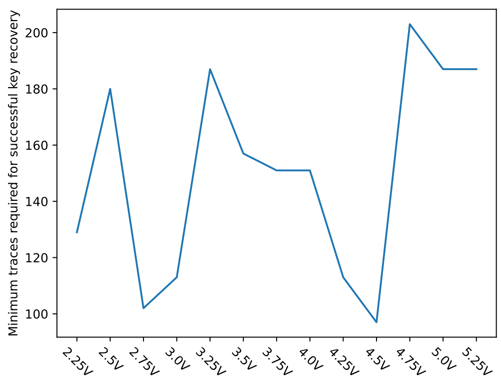


Figure 3.1 Relationship between operating Voltage and number of traces required for successful DPA

After the initial analysis of the data, no clear relationship was observed. However, this analysis has been rather generic and did not necessarily mean that there is no interesting pattern to be found. To be able to do further analysis effectively the amount of data had to be further reduced with the methods introduced in the section 1.4. These methods were experimentally verified on a random sample taken from the whole dataset. Under-sampling with factor 25 (the trace size is reduced 25 times) proved to be fairly efficient with surprising results – the minimal number of traces required to recover the key was almost four time less than that of the original traces for randomly chosen sample. The size reduction method which works with the knowledge of the hardware implementation was verified by comparing the distance from the peak the correlation with the source clocking signal which showed that the idea was correct. Although the latter method did not produce such extreme outcome for randomly chosen experiment data it produced more stable results corresponding to those of the initial analysis and as such was chosen for further use.

The Heatmap in figure 3.2 presents the minimal number of required traces (key recovery threshold) for each byte of the key, across different supply voltages. The minimal number of traces was calculated by creating a batch witch zero traces, when DPA failed recovering the correct key, new trace was added, this process was then repeated until DPA was successful for at least 30 newly added traces. The thresholds for each byte of the key across different supply voltages were within a relatively narrow range with no clear relationship between them. However, a high difference in the thresholds can be observed across different positions in the encryption key, which warranted further investigation. While the initial assumption was that the Hamming Weight would be an influential factor, the results as seen in the figure 3.3 suggest that there may be other

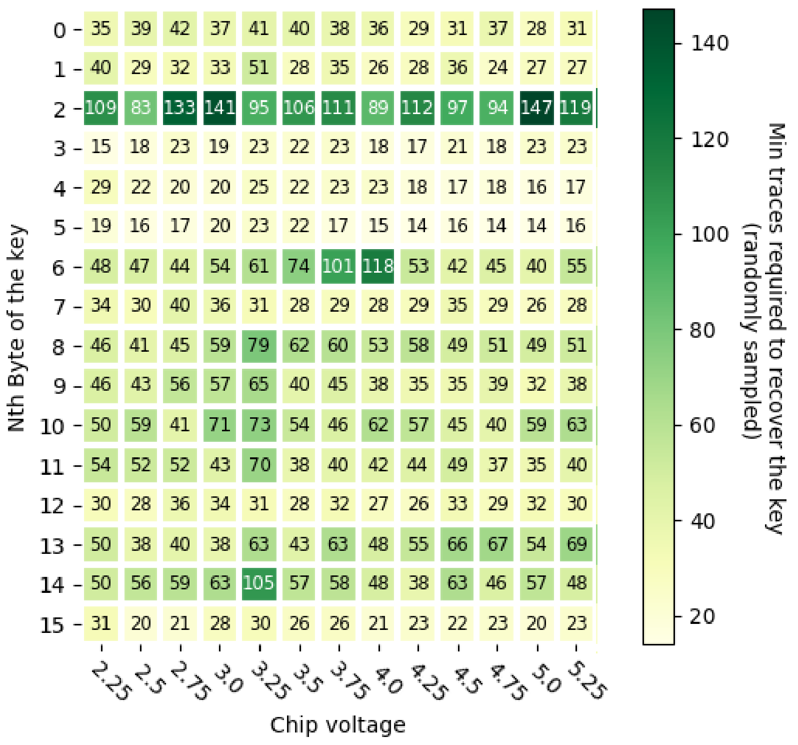


Figure 3.2 Minimal number of traces required for successful recovery of each key byte.

factors at play which are currently unknown and would require further work to be done.

As a final and last step in the analysis process, Partial Guessing Entropy (PGE) for different number of traces was calculated. The figure 3.4 shows the mean PGE value for all bytes. Again, the results show that there is no significant relationship between the supply voltages and PGE values. Notably, slight trend is observable within the range of 40 to 60 traces, in this section the PGE values indicate that the upper range of the operating voltage (3.75 V - 5 V) may leak slightly more information than the rest. However, the magnitude of the difference is not significant enough to produce a distinguishable impact on the overall success rate of the DPA attack.

3.1 Summary of the results

The analysis revealed no apparent relationship between the operating voltage of the target microcontroller and the success rate of the DPA attack. Coincidentally irregular pattern was observed in the success rate across different bytes of the key, but the available data did not reveal any apparent cause, leaving this area for further research.

Earlier proposed method for trace size reduction, based on the microcontroller clock signal, showed to be fairly efficient without causing any significant loss of information to the signal and has been used successfully in practice.

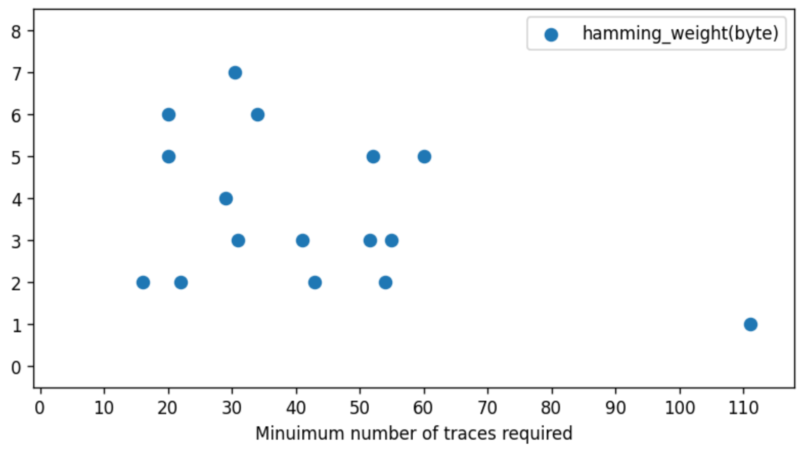


Figure 3.3 Relationship between Hamming Weight and required number of samples

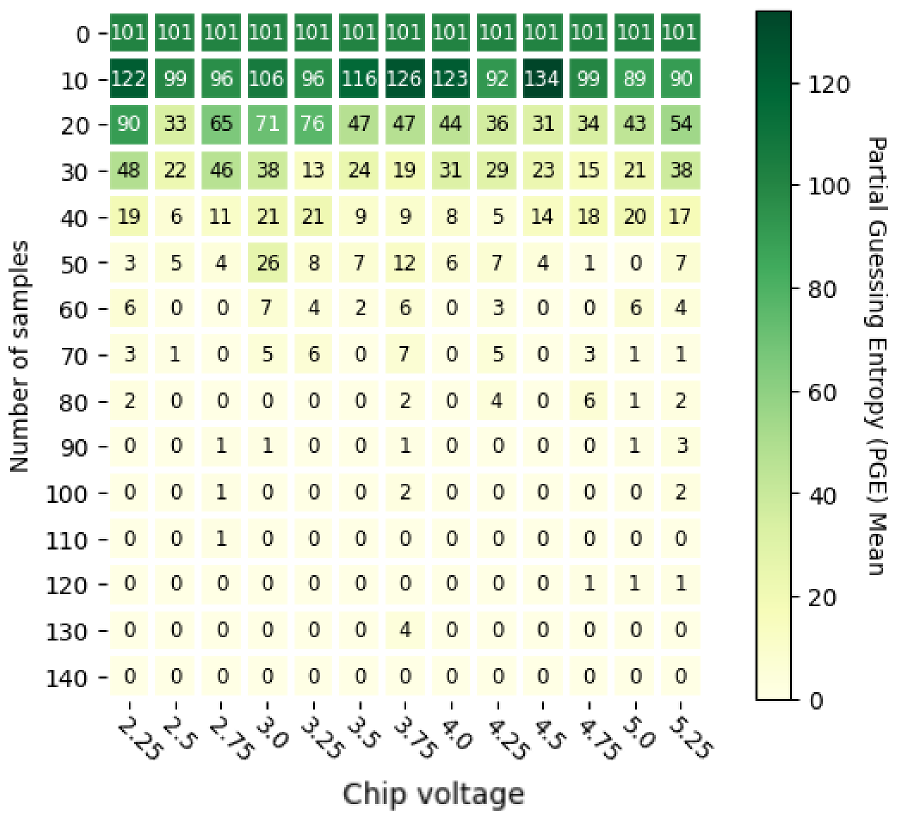


Figure 3.4 Values of Partial Guessing Entropy (PGE) for different number of traces and operation voltages.

Chapter 4

Conclusion

This work has explored the possibilities of Differential Power Analysis (DPA) as a method for attacking secure devices. Usage of QCoDeS library for data acquisition and capture has been demonstrated. DPA attacks were successfully and repeatedly launched against Atmega microcontroller. And the relationship between success rate of DPA and supply voltage of a target microcontroller has been examined.

Differential power analysis as a method to exploit side-channel data leakage unforeseen by Cryptosystem designers has been introduced. Variation of DPA called Correlation Power Analysis (CPA) has been explained as well and used extensively later in the work. Signal processing techniques relevant to the application of DPA have been discussed, some of them were experimentally verified and applied during the analysis process.

In order to communicate with the measurement equipment, namely *Keysight DSOX3024T* digital oscilloscope and *Aim-TTi QL355TP* lab DC power supply, the popular opensource data acquisition framework *QCoDeS* has been extended. The operation and usage of this framework was explained and illustrated with code examples. Some minor issues were encountered when storing data into the experiment database, which were most likely caused by the operating system, but they did not lead to any significant data loss.

14 sets of measured data have been acquired across different supply voltage levels ranging from 2.25 V up to 5.25 V. Voltage levels outside of the microcontrollers normal operating range were included as well in order to examine if the microcontroller behaves abnormally in extreme conditions. The entire traces have not been saved since the required storage space on the hard drive would exceed tens of gigabytes. Instead, only the important part containing the first round of AES was stored for later use.

In the last part of the work, previously recorded data was used to examine the relationship between voltage of the power supply and success rate of DPA attack. Firstly DPA attack has been launched against subsets of the raw recorded data, as this operation was both time and space consuming it was optimized by using binary search to find the smallest amount of traces required for successful full-key recovery. The recorded numbers have shown no correlation with the supply voltage and thus detailed analysis has been done in order to gain confidence in the conclusion. Then minimal required traces were calculated for each byte of the key using the processed traces in order to save computational time, which showed no relationship between attack success and supply voltage. However, the generated graph revealed interesting inequality among different bytes in the key. To further investigate this phenomenon, it would be necessary to examine traces recorded with different keys, unfortunately, such data was not available at the time of analysis, and thus this area remains open for future research. And the last part of the analysis uses Partial guessing entropy (PGE) to measure information leakage as this method is less prone to small variations in the success. As with the previous two instances, no relationship between supply voltage of the microcontroller and DPA success rate was found and therefore the following conclusion can be drawn.

Success rate of the DPA attack mounted against Atmel Atmega8P microcontroller is independent on the voltage level used to supply power microcontroller.

Chapter 5

Future work

5.1 Success rate and encryption key format

Figure 3.3 revealed inequality between different bytes of the key. Different number of traces was required to recover different bytes, various factors can be at play, such as Hamming Weight of the key. Data which would support or contradict the assumption were not available at the moment. Trace recordings made with different encryption keys are required in order to analyze the relationship.

5.2 Secure hardware

Different protections against DPA have been proposed in research papers such as Dual-rail pre-charge logic (DPL), Random delays insertion (RDI) or Bus-Invert Coding. DPA success rate against devices employing such counter measures might possibly be influenced by the supply voltage, but further research is required.

DPL uses two rails in order to represent one logical bit of value and the computations are done in two phases – pre-charge and evaluation. [26]

RDI works by inserting random delatys into the datapath of a cryptographic processor randomizing both the consumption profile as well as the total charge quantity transferred from the power supply. [27]

Bus-Invert Coding aims at reducing the number of bit transitions in a circuit. When the number of expected transitions is larger than threshold HW, the input is coded in order to reduce the number of transitions required. [28]

5.3 DPA targeting the last encryption round

As mentioned in the chapter 1.1 DPA attack can target two different parts of the AES encryption. In our work the intermediate values after the first subBytes operation have been targeted. Future work could verify the results acquired by targeting the value of the roundKey used in the last round. The process of deriving round keys is reversible and therefore the value of the encryption key can be calulated.

Bibliography

1. KOCHER, Paul; JAFFE, Joshua; JUN, Benjamin; ROHATGI, Pankaj. Introduction to differential power analysis. *Journal of Cryptographic Engineering*. 2011. Available from DOI: `10.1007/s13389-011-0006-y`.
2. KOCHER, Paul; JAFFE, Joshua; JUN, Benjamin. Differential Power Analysis. In: *Advances in Cryptology — CRYPTO' 99*. Springer Berlin Heidelberg, 1999. Available from DOI: `10.1007/3-540-48405-1_25`.
3. BRIER, Eric; CLAVIER, Christophe; OLIVIER, Francis. Correlation Power Analysis with a Leakage Model. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 16–29. Available from DOI: `10.1007/978-3-540-28632-5_2`.
4. BUČEK, Jiří. *BI-HWB support material* [online]. 2023. [visited on 2023-02-15]. Available from: `https://courses.fit.cvut.cz/BI-HWB/`.
5. BOTTINELLI, Paul; BOS, Joppe. Computational aspects of correlation power analysis. *Journal of Cryptographic Engineering*. 2017. Available from DOI: `10.1007/s13389-016-0122-9`.
6. NOVÁK, Petr. *BI-PST support material* [online]. 2023. [visited on 2023-02-15]. Available from: `https://courses.fit.cvut.cz/BI-PST/`.
7. *DPA Contest* [online]. [N.d.]. [visited on 2023-03-19]. Available from: `https://www.dpacontest.org/v2/index.php`.
8. DAOR, Joa; DAEMEN, Joan; RIJMEN, Vincent. AES proposal: rijndael. 1999.
9. ISO CENTRAL SECRETARY. *Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers*. Geneva, CH, 2010. Standard, ISO/IEC 18033-3:2010. International Organization for Standardization. Available also from: `https://www.iso.org/standard/18033.html`.
10. *SciPy: Scientific Library for Python* [online]. 2023. [visited on 2023-03-14]. Available from: `https://www.scipy.org/`. Version 1.10.1.
11. MANGARD, Stefan; OSWALD, Elisabeth; POPP, Thomas. *Power Analysis Attacks*. Springer US, 2007. Available from DOI: `10.1007/978-0-387-38162-6`.

12. WOUDENBERG, Jasper G. J. van; WITTEMAN, Marc F.; BAKKER, Bram. Improving Differential Power Analysis by Elastic Alignment. In: *Topics in Cryptology – CT-RSA 2011*. Springer Berlin Heidelberg, 2011, pp. 104–119. Available from DOI: `10.1007/978-3-642-19074-2_8`.
13. ELAABID, M.; GUILLEY, Sylvain; HOOGVORST, Philippe. Template Attacks with a Power Model. *IACR Cryptology ePrint Archive*. 2007, vol. 2007.
14. *ATmega8(L) datasheet* [online]. Atmel Corporation, 2013. [visited on 2023-03-01]. Available from: `https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf`. Rev.2486AA–AVR–02/2013.
15. NIELSEN, Jens Hedegaard. *QCoDeS/Qcodes: QCoDeS 0.37.0*. Zenodo, 2023. Available from DOI: `10.5281/ZENODO.7573346`.
16. GRECCO, Hernán E.; DARTIAILH, Matthieu C.; THALHAMMER-THURNER, Gregor; BRONGER, Torsten; BAUER, Florian. PyVISA: the Python instrumentation package. *Journal of Open Source Software*. 2023, vol. 8, no. 84, p. 5304. Available from DOI: `10.21105/joss.05304`.
17. *LMx58-N Low-Power, Dual-Operational Amplifiers* [online]. Texas Instruments Incorporated, 2000 Revised March 2022. [visited on 2023-03-01]. Available from: `https://www.ti.com/lit/ds/snosbt3j/snosbt3j.pdf?ts=1682409152028`.
18. *AVR109: Self Programming* [online]. Atmel Corporation, 2004. [visited on 2023-03-01]. Available from: `https://ww1.microchip.com/downloads/en/Appnotes/doc1644.pdf`. Rev. 1644G–AVR–06/04.
19. DEAN, Brian S. *AVRDUDE 0v7.1* [online]. 2023. [visited on 2023-02-20]. Available from: `https://github.com/avrdudes/avrdude`.
20. BUČEK, Jiří [private communication]. 2023-02-27.
21. *PyCryptodome* [online]. 2023. [visited on 2023-03-14]. Available from: `https://github.com/Legrandin/pycryptodome`. Version 3.17.0.
22. *QL Series II - Precision Power Supplies - Instruction Manual* [online]. Thurlby Thandar Instruments Ltd, 2013. [visited on 2023-03-01]. Available from: `https://resources.aimtti.com/manuals/QL_Series_II-Instruction_Manual-Iss8.pdf`. Rev. 48511-1560 Issue 8.
23. LXI CONSORTIUM, INC. LXI Reference Design Overview [online]. 2016 [visited on 2023-03-19]. Available from: `https://www.lxistandard.org/Documents/Papers/LXI%5C%20Reference%5C%20Design%5C%20Overview%5C%2016MAR2016.pdf`.
24. *InfiniiVision 3000T X-Series Oscilloscope* [online]. Keysight Technologies, 2022. [visited on 2023-03-01]. Available from: `https://www.keysight.com/us/en/assets/7018-04570/data-sheets/5992-0140.pdf`.
25. *InfiniiVision 3000T X-Series Oscilloscope - Programmer's Guide* [online]. Keysight Technologies, 2022. [visited on 2023-04-01]. Available from: `https://www.batronix.com/files/Keysight/Oszilloskope/3000XT/3000XT-Programming.pdf`.

26. DANGER, PJean-Luc; GUILLEY, Sylvain; BHASIN, Shivam; NASSAR, Maxime; SAUVAGE, Laurent. *Overview of Dual Rail with Precharge Logic Styles to Thwart Implementation-Level Attacks on Hardware Cryptoprocessors*. 2009. Tech. rep. Available also from: `https://hal.inria.fr/inria-00075774`. hal-00431261.

27. BUCCI, Marco; LUZZI, Raimondo; GUGLIELMO, Michele; TRIFILETTI, Alessandro. A countermeasure against differential power analysis based on random delay insertion. In: 2005, 3547–3550 Vol. 4. Available from DOI: `10.1109/ISCAS.2005.1465395`.

28. VOSOUGHI, M. Ali; WANG, Longfei; KOSE, Selcuk. Bus-Invert Coding as a Low-Power Countermeasure Against Correlation Power Analysis Attack. In: 2019. Available from DOI: `10.1109/SLIP.2019.8771332`.

Contents of enclosed Media

- `readme.txt` ... file with contents description
- `src`
  - `mega8aes.zip` ... project bundle with Atmega firmware
  - `DPA` ... Directory containing implementation sources
    - `AimTTiPatched.py` ... QCoDeS driver for QL355TP power supply
    - `KeysightInfiniiumPatched.py` ... QCoDeS driver for keysight infiniium oscilloscopes
    - `Measurement.db` ... Database containing measured traces
    - `Measurement.ipynb` ... Jupyter notebook containing the Measurement and analysis
    - `requirements.txt` ... Python package requirements file
    - `utils.py` ... various utility functions
  - `thesis` ... directory containing the LaTeXsource files
- `text` ... directory containing thesis text
  - `thesis.pdf` ... the thesis text in PDF format