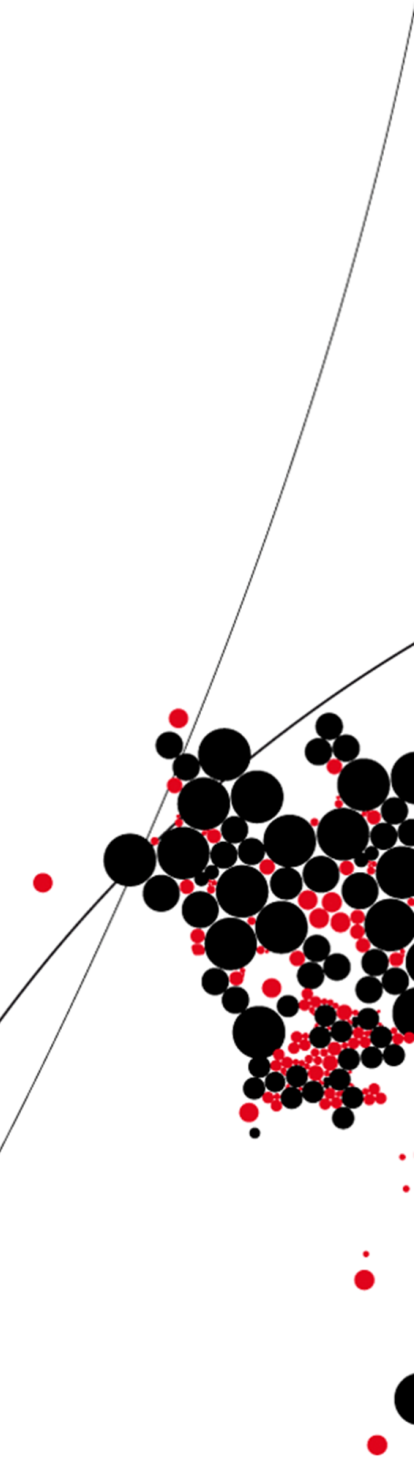
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

Applying nucleotide sequence alignment techniques to side channel analysis

Heitor Uchoa

Msc. Thesis
February 2023



Committee:
dr. ir. N. Alachiotis
dr. ir. M. Ottavi
dr. ir. A. Continella
msc. ir. V. Arora

Computer Architecture for Embedded Systems (CAES) Group
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

It is well known in the embedded security field that the power consumed by a device depends on the operations and data that they process at that moment in time. This information can leak and help to extract secret information such as cryptography keys. Protecting devices from these attackers requires Side Channel Analysis (SCA) countermeasures that add randomness to the power traces, making exploiting leakages challenging.

Alignment problems are also common in genetics, where they use alignment methods to study biological relationships between species and ancestors. This work presents a new method for the alignment of power traces that implements ideas from bioinformatics and shows more effective alignment by using fewer power traces to extract cryptography keys than existing methods.

Contents

List of acronyms

SCA	Side Channel Analysis
MSA	Multiple Sequence Alignment
DNA	Deoxyribonucleic acid
RNA	Ribonucleic acid
DES	Data Encryption Standard
AES	Advanced Encryption Standard
IP	Initial permutation
FP	Final permutation
DPA	Differential Power Analysis
CPA	Correlation Power Analysis
GUI	Graphical User Interface
SW-DPA	sliding window Differential Power Analysis (DPA)
DFA	Differential Fault Analysis
MAFFT	Multiple Alignment Fast Fourier Transform
SaX	Symbolic Aggregate Approximation
PAA	Piecewise Aggregate Approximation
DTW	Dynamic Time Warping
RAM	Rapid Alignment Method
rdm	Relative distinguishing margin
FPGA	Field-programmable gate array

Chapter 1

Introduction

Nowadays, it is common to depend on embedded devices that process private information about us. This information has to be hidden from other parties, preventing it from being extracted by adversaries. Furthermore, they must also exchange this information with other devices through some network. These devices could be anything: smart cards used as credit cards, our phones, or even cars. As a solution to maintain the secrecy of this information, cryptography plays a significant role. This approach relies on encrypting plain text to ciphertext using a key. A device transmits a ciphertext to another that can decrypt it back to meaningful information using a proper decoding key. When systems use the same key, that is called symmetric cryptography, which is the focus of this work.

Although these systems have solid and reliable implementations, there are studies on extracting secret information of cryptography systems vulnerabilities using side channels. An attacker can either use analytical attacks([1], [2], [3]) or non-intentional channels to extract this information. Non-intentional channels exist because these systems use electronics/electricity to operate on information while performing an algorithm. These channels can be power consumption, electromagnetic radiation, sound, temperature signatures, and time. All of these are called ”Side Channels”. There are examples available in the literature([4] [5] [6]). Processing the side channel leakage to obtain useful information is called Side Channel Analysis (SCA).

SCA is possible since electronic devices operate over data that is either zero or one, and most systems represent these as 0 volts (low or 0) or 3.3 / 5 volts (high or 1). Thus, more bits set to one, more power consumption. Techniques such as Differential Power Analysis (DPA) [5] and Correlation Power Analysis (CPA) [7] use power variations to reveal a device cryptography key. These techniques are well known in the field of SCA and are used to determine vulnerabilities while developing systems or software or by attackers to extract data used by devices. Extracting data from these power traces requires acquiring multiple power traces from the same device

while performing the same operation on random inputs and capturing its outputs. There is less data leakage when traces are misaligned. The misalignment reasons can be as simple as variations on triggering times to more complex implementations that aim to cause misalignment. The alignment of the same operations in time, maximizes data leakage of power traces during data-dependent operations.

As an approach to increasing the difficulty of the attacks, cryptography systems implement countermeasures. These countermeasures target making alignments a challenging task by adding randomness to power traces. Some approaches are: adding noise to the power traces and introducing countermeasures as random operations. Random operations generate misalignment in multiple points of the power traces. Unstable clocking or random delays [8] are options to achieve misalignment. Countermeasures are mutations between power traces from the same sequence. Identifying these mutations will be required to align these traces. Once that is done, the aligned of leaky zones may provide secret information.

There are different reasons for misalignment, not all caused by countermeasures. Some of them come from simple inaccuracies when measuring the power traces. Static alignment [4] is an option to align trace sets that contain misalignments caused by these inaccuracies. This method chooses a reference trace and shifts the other traces based on this reference point. In the presence of countermeasures that add randomness in the time domain, methods like Elastic alignment [9], or sliding window DPA (SW-DPA) [8] can be useful alternatives; these also use a reference trace to perform an all to one alignment.

However, proper alignment and mutation identification are not only important in the SCA field. Another field where alignments are crucial is bioinformatics, specifically in the field of genetics, to understand Genomes. Genomes are organisms' genetic information formed by nucleotide sequences that determine their characteristics. In the case of a virus, these are called Ribonucleic acid (RNA). The sequences of nucleotides form Deoxyribonucleic acid (DNA), RNA, or proteins. Understanding evolutionary relationships, homology, and mutations between viruses or species require understanding these sequences. A common method in bioinformatics is called MSA [10]. This method is the option in case of three or more sequences need alignment.

Aligning power traces is normally done by finding a reference trace and performing a pairwise alignment of all traces with that specific trace, sometimes modifying the traces by elongating the trace based on the reference trace and modifying the original data. MSA performs an all-to-all alignment. This alignment method adds gaps to shift parts of the sequences, creating new regions of low/high similarities. The method does not lose data as the previous approach; this could mean an improvement to SCA.

The work develops a novel approach on SCA alignments by translating the ideas used by MSA in bioinformatics to this field, proposing a new alignment option when countermeasures are in place.

1.1 Motivation

Exploiting leakages to break cryptography keys can be challenging. Countermeasures implementation increases the effort required to exploit leakages when attackers use existing methods, such as static [4] and elastic [9] alignments. This work investigates a novel approach to alignment for SCA, aiming to provide a new method that works in the presence of countermeasures. This idea investigates how to translate the knowledge from the field of bioinformatics to SCA field, more specifically, on the field of genetics. In bioinformatics, alignments are also used to understand relationships between species and their mutations over time by alignment of sequences of DNA, for example.

The existing algorithms mentioned in this section use a reference trace to modify the other traces in the trace set. Thus, increasing similarity with the reference trace and possibly losing information when averaging or interpolating samples. This work is motivated to use MSA as a novel method where an all-to-all alignment is done and investigates the results by asking the research questions posted in the next section.

In terms of data safety intentions, this work falls in a midway zone in terms of offense and defense to cryptography systems. The goal is to understand MSA as a pre-processing step in an attack to extract information. By doing so, providing information to prevent this attack.

1.2 Research Questions

As previously stated, these fields align sequences to understand their relationship and extract useful information. These both also need to avoid misinterpretations caused by mutations/countermeasures. This research aims at the mentioned shared goal and asks a main question:

- **How can multiple sequence alignment be used for side channel analysis alignments to make side-channel attacks more effective?**

Sub-questions The following sub-questions targets opening discussion and divide the main questions into steps that could provide more direction in finding the final goal:

- **Which is the most appropriate performing multiple sequence alignment tool for this research regarding the number of samples and sequences to be aligned?**
- **What are the options to convert traces data sets into nucleotide sequences for multiple sequence alignment?**
- **If countermeasures are in place, How is it possible to minimize their effect on side-channel analysis/attacks using multiple sequence alignment?**
- **How this alignment compares to existing alignments such as static and elastic?**

The sub-questions are in place to aid the search for the main question by breaking it into smaller problems. The first sub-question calls an investigation of the MSA methods as different options vary on-time performance, accuracy, length, and the number of sites of the sequences. Since MSA will be applied here to power traces, it is important to understand which method is more appropriate to the given sequences, given their length and number of sites (samples, in the case of power traces). The second question targets investigating the conversions from data sets of power traces to ”nucleotides”. The challenge investigated by this question lies in the fact that multi-sequence alignments use a limited number of symbols (20 characters for proteins and around 9 for DNA/RNA), and samples are bytes that have 256 values. The next question challenges the method's performance when attacking a device with countermeasures on its traces. Suppose it is possible to minimize their effect and perform an attack: How does this method compare to the attack success of existing and commonly used approaches?

1.3 Report organization

This report follows the structure: Chapter 2 discusses the background of this work explaining in detail Side Channel Analysis (SCA), the background on cryptography algorithms, and Multiple Sequence Alignment (MSA). Chapter 3 shares related work on the field of SCA and MSA. It also included work in the field of time series representation, as this is a big part of this work. Chapter 4 discusses how this work implementation, and its different methods. The following chapter 5 gives an overview of the technologies used for this implementation. Chapter 6 discusses the results of the work in many comparisons, followed by conclusions and discussions over chapter 7.

Chapter 2

Background

This chapter presents and discusses the background for understanding this work. The first step is understanding cryptography, what is SCA, and how to perform it. That should lead to understanding why to use MSA.

2.1 Cryptography

Understanding this work requires understanding, at least superficially, how encryption algorithms work. In short, encryption is transforming a text defined as plain text to ciphertext; the goal is to hide information from unwanted readers/attackers before transmitting it through a public channel. Encrypting plain text or decrypting requires using a key, and when this key is the same for both sender and receiver, this algorithm is a symmetric-key algorithm. If the algorithm uses different keys, it is known as an asymmetric key. This section introduces AES and DES algorithms.

2.1.1 DES

Data Encryption Standard (DES) is a symmetric-key algorithm containing steps called encryption rounds (16 rounds/blocks). Every round uses a different key derived from a 64-bit key. Important to notice that the key is 64 bits, where 8 bits are parity checking, and the remaining 56 bits generate the sub-keys. The block size is 64 bits, and the sub-key size is 48 bits for each of these blocks, and it follows the Feistel Ciphertext structure. This method is less used nowadays as many other works have already proven that it is not secure; despite that, this is a didactic algorithm, and it facilitates the understanding as a study case.

The fig. 2.2 shows the structure of the DES algorithm, its step by step is described below:

1. Initial permutation (IP) is performed on the input

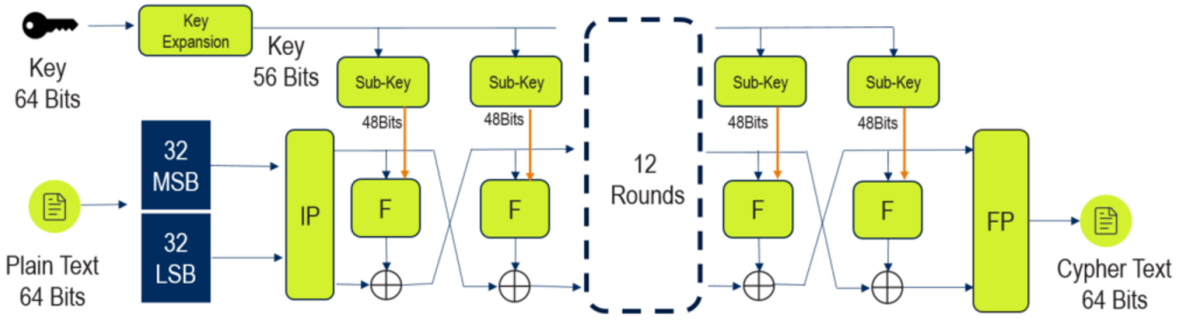


Figure 2.1: DES Top-level representation

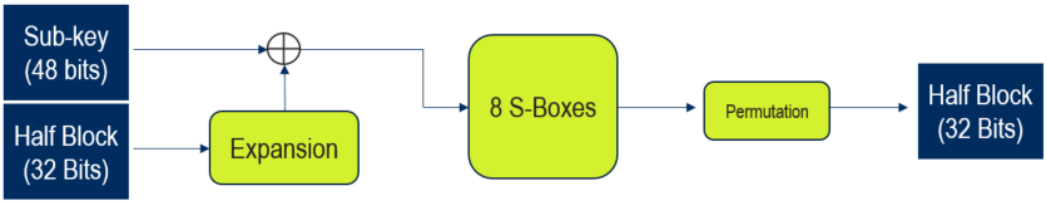


Figure 2.2: Feistel function

2. Feistel function used on Right (32 bits) with sub-key n = 1

3. Previous result is XOR'ed with the left (32 bits)

4. Left block (32 Bits) and right block (32 Bits) are swapped; the right block becomes the left block, and the left block becomes the right block.

5. Increase sub-key n = n + 1 and repeat from item 2 again, 15 times.

6. Final permutation (FP) (IP Inverse operation)

In reality, The final and initial permutations have no significance to cryptography, but hardware limitations forced its need during its development time. At each round, the Feistel function operates on one of the 32 bits blocks. It is important to understand the Feistel function to perform side-channel analysis in this cryptography. It consists of four steps:

1. Expansion

2. XOR (Data and Sub-key)

3. Substitution (S-Boxes)

4. Permutation

In the first step, the algorithm expands 32 to 48 bits. These 48 bits are each of the four initial bits together with their neighbor bits from both sides, resulting in 6 bits * 8 (48 bits). These 48 bits and the sub-key are inputs to an XOR function. The result of the XOR function is divided into 8 S-Boxes, each returning a 4 bits result. These S-Boxes function as look-up tables, and they are non-linear operations. In the last step, the S-boxes output goes through a permutation function that aims to spread the output, so the bits go through different S-Boxes on the next round. Fig. 2.7 shows an example of this transformation. Although other encryption algorithms use it, the idea is similar.

2.1.2 AES

Since DES is not safe anymore, and AES [11] comes as the safer option. This algorithm does not use the Feistel function as the previously discussed implementation. It uses 128 bits block size and key sizes of 128, 192, or 256 bits. The rounds vary accordingly, 10, 12, or 14, respectively. The AES specification [11] has the following steps for encryption on its input:

1. Create State Array
2. Add round key
3. Bytes Substitution
4. Shift rows
5. Mix columns
6. Add round key (Repeat from item 2 more 9,11 or 13, key length dependent)
7. Bytes Substitution
8. Bytes Substitution
9. Shift rows
10. Add round key

The AES operates on a four rows matrix of bytes called a state array. The columns' quantity comes from the block length mentioned previously, divided by 32 bits. Figure 2.3 shows how a block of 128 bits becomes a state array. All AES operations perform transformations on this state array and output the ciphertext in the same approach. Add Round key operations(step 2) adds the round key to the State array. This is an XOR operation done in a column-by-column manner as per figure 2.6.

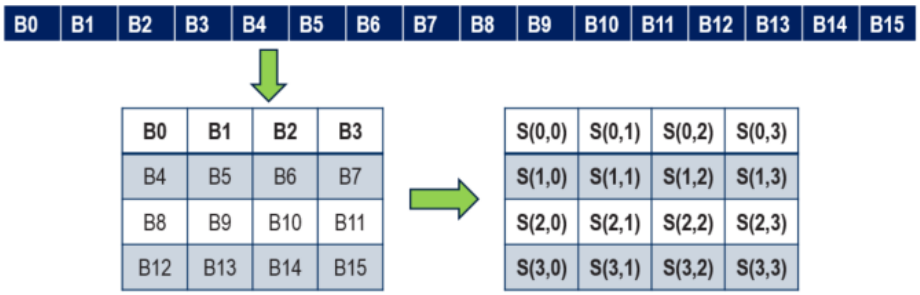


Figure 2.3: AES state array

The Shift rows operation is done by cyclic rotating each row to the left by a certain number. Every row shifts by the number that refers to its index, assuming the first row is zero. See figure 2.5.

The mix columns step operates on a column-by-column approach as the Add round key step, See figure 2.4. The transformation treats the columns as a four elements polynomial. The bytes substitution is similar to what DES does; this is a non-linear transformation and operates at every State matrix index. Knowing how to use it and how to perform the substitution is essential. How to create the S-Box look-up table 2.7 does not affect the understanding of this work. The S-Box transforms every index in the state array. Understand the most significant 4 bits as X and the

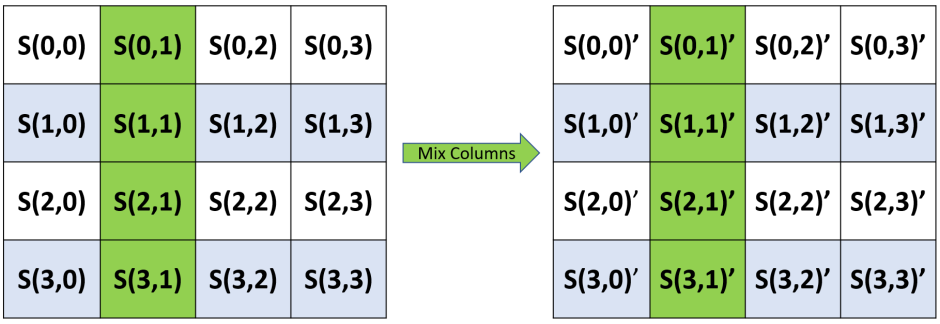


Figure 2.4: AES Mix round

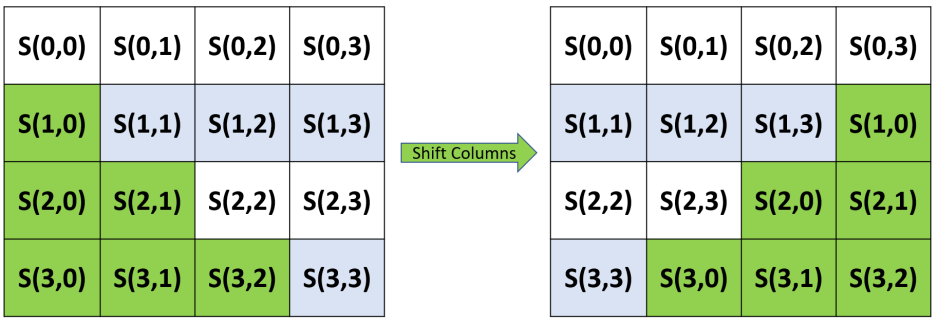


Figure 2.5: AES Shift Round

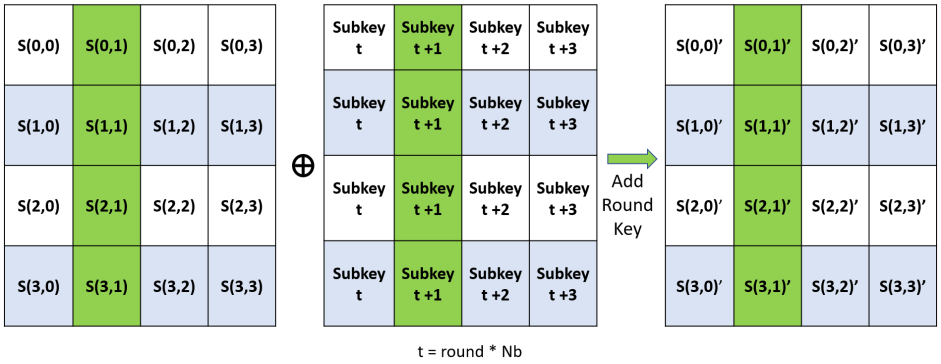


Figure 2.6: AES Add round

least significant bit as Y and use the table from figure 2.7 to generate the result. For example, if the value of an index is ”aa” in hexadecimal, its result is ”ac”. In the example, there is a transformation from hexadecimal ”FC” to ”B0”.

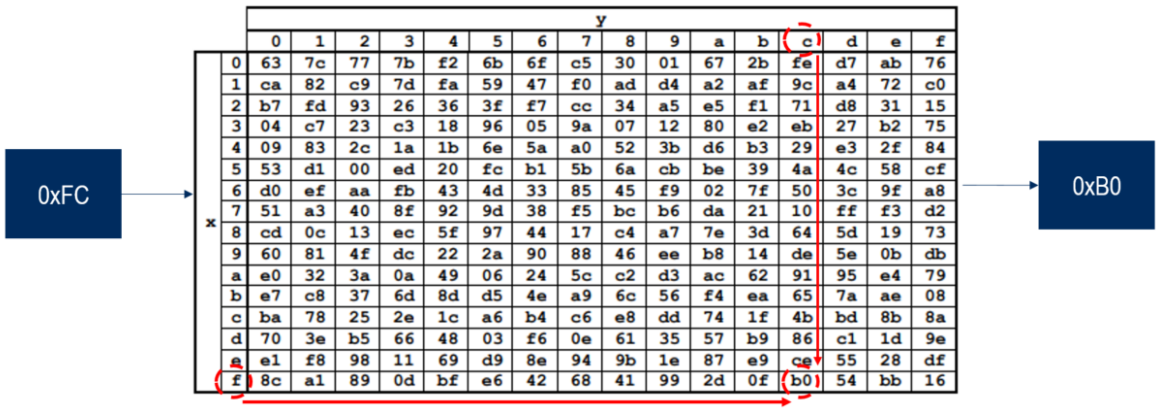


Figure 2.7: AES S-Box

The key Expansion step comes before the add round key to generate a key schedule [11]. For this work, it is important to understand that the key schedule derives from the Ciphertext key, and every round uses it.

2.2 Side Channel Analysis

Side Channel Analysis (SCA) is the field that uses information from the called Side channels. These are channels where information can leak through, given either a failure in implementation planning or just the natural way how computational systems function. Side channels can be temperature, power consumption, sound, and more. For example, the computer using an editor application to write this document consumes less power than the same computer running a last-generation game and

heats less. Meaning that just by the amount of heat the computer generates, it is possible to exploit that it is running a heavier application. Side channels are channels through which a device or application leaks unintended information. These can be temperature, sounds, power consumption, electromagnetic radiation, or time. Leakages analysis can lead to breaking the cryptography keys when applying leakages models at the crypto operation time. Extracting useful information from unintended leakages is called Side Channel Analysis (SCA). A side channel is a channel through which unintentional information leaks.

2.2.1 Power Traces

This work focuses mainly on SCA of power traces such as the ideas discussed here can be applied to other side channels such as electromagnetic radiation. On first look, when investigating power traces, it is possible to observe patterns in them when they have multiple repeated operations. Let us use the DES operation as an example. Sixteen rounds of the same operations perform transformation over the data input. Fig. 2.8 shows this procedure; exactly 16 rounds appear.

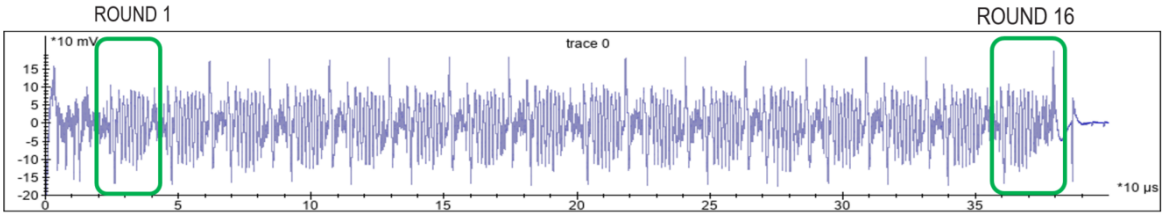


Figure 2.8: DES Power trace

These traces come from devices performing cryptography operations such as DES or AES. An oscilloscope needs a trigger signal to start recording precisely at the start of the target operation; the computer stores this information. This computer also inputs data to the device and receives the encrypted output data after operations. This specific power trace has its input and output linked to it. Figure 2.9.

A device has four types of power consumption :

1. Constant
2. Electrical noise
3. Data dependent
4. Operation Dependent

The first type is irrelevant to our investigation but represents an offset to the power trace. The second one, however, can interfere with analysis if not minimized.

The critical aspects for a SCA are the data and operation-dependent powers. Electronic devices use different voltages to represent data, so their power trace contains information about the data and the operation that it processes. In that sense, an operation of 2 + 2 and 7 + 7 will have a similar shape in the power traces. However, amplitudes can differ based on data values. Since 2 and 7 represented in binary are ”0010” and ”0111”, respectively. Bits set to ”0” in a hardware system are 0 volts, and the ”1” representation uses 3-5 volts; thus, more bits active to represent data imply more power consumption, which affects the amplitude in our measurements. SCA extracts information that variations between two power traces of the same device will differ when it processes different data.

2.2.2 Trace Alignment

Data leakage can exist when multiple power traces are aligned, and a data-dependent operation happens at a given time for these multiple traces. Good trace alignment is crucial for attacks as DPA([5]) or CPA([7]) possible. The next sections explain these cryptography attacks on embedded devices. For now, let us keep in mind that they need multiple traces containing regions performing the same operation over different data aligned aiming to make use of the variation observed in fig. 2.11.

The attacks mentioned here are well-known in the industry. Therefore producing cryptography hardware or software implementations that are safe from these attacks is a goal. Thus, developers create countermeasures to interfere with the power traces so that they do not convey information about what is happening during its operation.

1. Random Delays

2. Clock Jitter

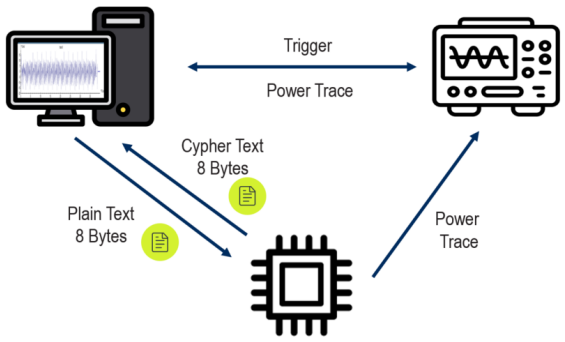


Figure 2.9: SCA Physical setup

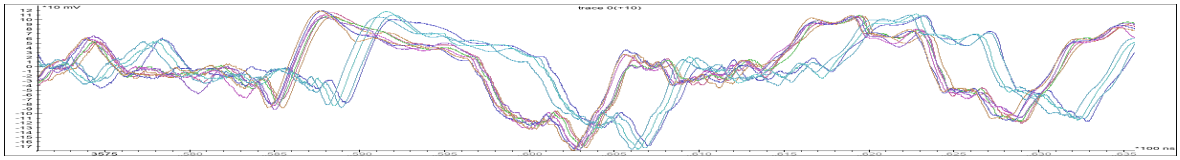


Figure 2.10: Non-aligned power trace

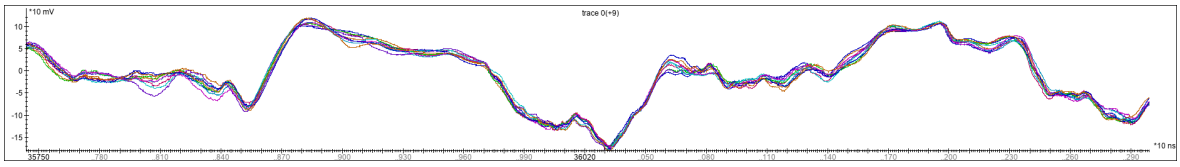


Figure 2.11: Aligned power trace

2.2.3 Countermeasures

Countermeasures aim to extinguish or mitigate vulnerabilities to attackers using the methods discussed so far and available in the literature [12]. These countermeasures come from hardware or software implementations. Hardware implementations that add noise or non-Gaussian noise to the power traces are available in the literature [13]. Although it is essential to mention these countermeasures, this work focuses on ones that create process desynchronizations by inserting random anomalies into the power traces. These anomalies are unstable clocking or random delays [8]. The main idea is to introduce disturbances to the side channel so that it is misaligned when compared to the same operations power trace, making it a challenging task for an attacker to exploit leakages. A practical example of random delays is in fig. 2.12 and 2.13, in these two pictures, the alignment of 10 power traces is attempted at two different regions. These traces come for the same device executing the same operation. Aligning at one of the regions leads to a complete misalignment in others; random delays were introduced as a countermeasure for this case. They minimize the possibility of extracting secret information through data leakage at that point.

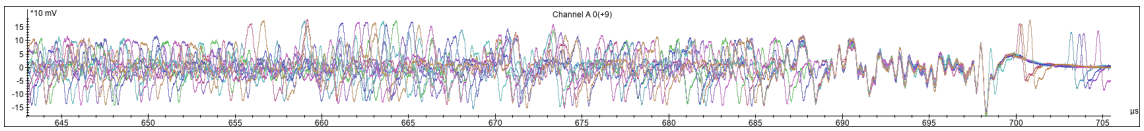


Figure 2.12: Power trace with random delays

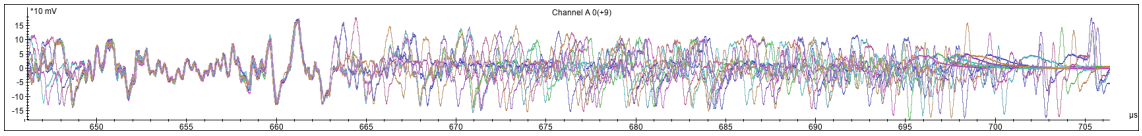


Figure 2.13: Power trace with random delays 2

2.3 Crypto operations power analysis

This work used three different methods to analyze the final alignments of power traces and to attempt to extract the cryptography operation sub-keys. These are CPA, first-order analysis, and know-key analysis. This section presents the knowledge required to understand them.

2.3.1 Correlation Power Analysis

DPA is a well know method in the field of SCA, this being introduced by Paul Kocher [5]. This work will widely use the method that came as a step further from the Paul Kocher method against crypto devices, the CPA [7]. This method relies on the fact that power depends on the number of ones an operation uses, and many attacks use this idea [14]. The quantity of bits set to '1' in a binary value is called Hamming weight. It formal definition can be considered as $H(N) = \sum_{n=1}^{m}(n_i)$, assuming the first index is 1. Fig. 2.9 represents the initial setup, and fig.2.14 shows more detail in its use. The idea is to use a setup that inputs random messages to the cryptography device and captures its output.

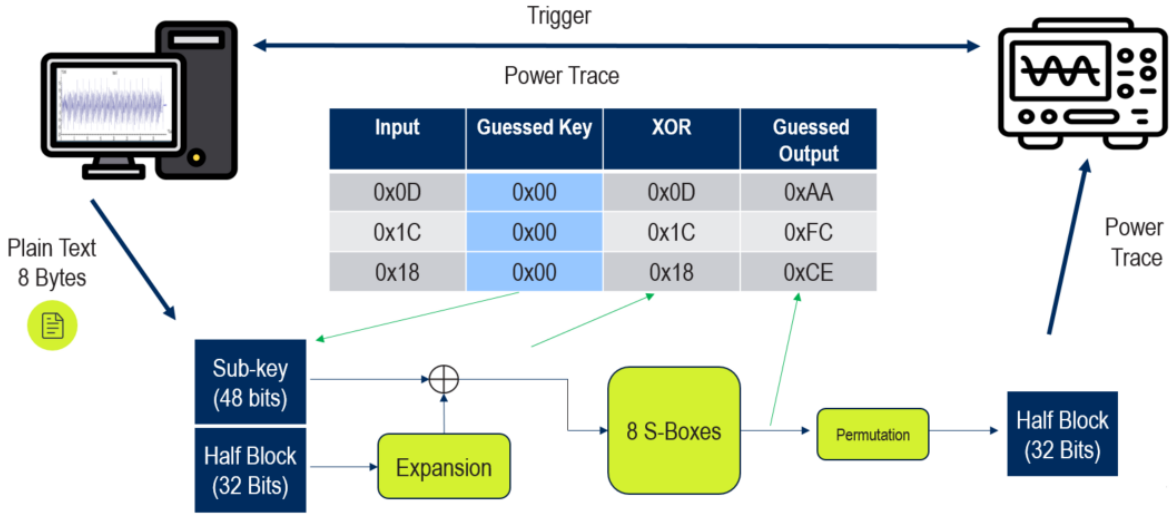


Figure 2.14: CPA hardware setup

At this point, to attack the cryptography operation, the sub-keys need to be

guessed. The sub-keys are smaller than the key itself, meaning it is much easier to make guesses and brute force. The operation is known as it is described by the DES or AES standards, as previously mentioned. The expected outputs are known for a specific guessed key. Although the guessed output is not correct, they can show the rank. The fig.2.14 shows specifically this idea for a DES Feistel function mentioned previously.

Pearson correlation of the power traces and the inputs or outputs review an estimate of the subkeys. The key with the higher correlation is determined to be the correct one.

The approach for the correlation here is to correlate the variation of the hamming weights from various inputs with the variation of an x-axis sample in all traces. If traces together are imagined as a matrix, the correlation of these hamming weights will be done per column resulting in one single vector. Fig. 2.15 shows the correlated data per trace and their correlation result where high peaks exist where this data is leaking.

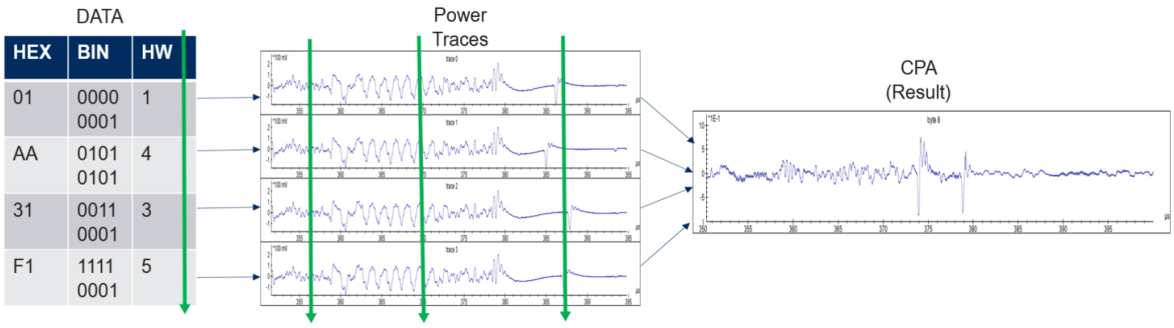


Figure 2.15: CPA Example

2.3.2 First order analysis

This method retrieves the key by using statistical analysis methods. It generates a list of candidates, their confidence, and their position, and it also implements Relative distinguishing margin (rdm) [15]. The better the alignment in the traces, the higher the output confidence for specific keys is. The method is implemented within the module from software that this work uses to evaluate final results. It was an intermediate step toward evaluating the algorithm proposed in this work. Once the alignment is complete, if this algorithm can find the keys for the cryptography operation, the known-key analysis provides more information about the results.

2.3.3 Known-Key Analysis

Known-Key Analysis is an evaluation module available in Riscure Inspector [16] and can give an interpretation of a set of power traces. It shows leakage strength related to every key byte at each data point of the input trace, and it can present a plot of this data. The know-key analysis evaluates this work results.

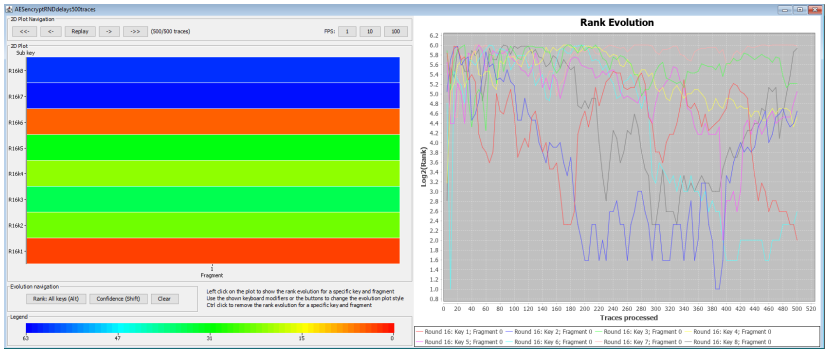


Figure 2.16: AES Known-key analysis : rank evolution

Fig. 2.16 is an example of what to expect. On the left side plot, the Y-axis represents the sub-keys of the ciphertext, and its X-axis represents all analyzed fragments. Note that each fragment accounts for all trace samples for this example. That is available as a parameter, and multiple fragments per sub-key are an option by choosing the number of samples to be represented (number of fragments = (total number of samples) / (fragment length)).

The colors in the plot represent the rank of the found key value based on the known-key value for that subkey. A rank of 1 means that the recovered value and the known key value for that subkey match. Hovering over a block on the plot shows the exact value of the rank. The plot on the right side shows the evolution of this rank of a specific subkey and fragment per number of traces.

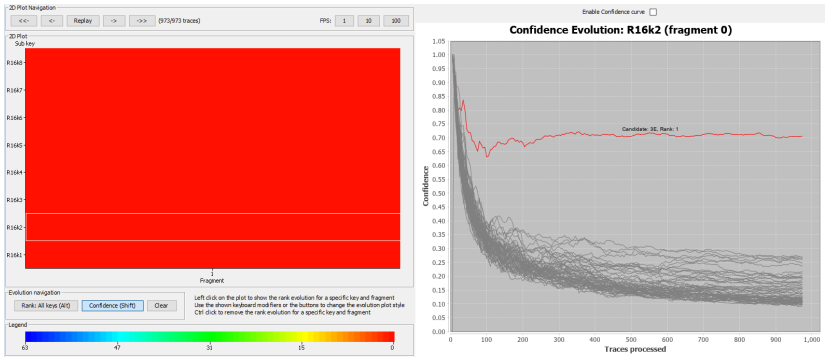


Figure 2.17: AES Known-key analysis : confidence evolution

Fig.2.17 also shows another plot option, an evolution plot of how the confidence of the known key value for the selected subkey relates to all the other values of that

subkey as more traces are analyzed. Expect a successful attack, and the correct value will separate itself from the others as the key in the figure has separated itself.

2.4 Nucleotide Sequence Alignment

A nucleotide is the building block of nucleic acids such as DNA and RNA. These two have a specific sequence of nucleotides arranged in a particular order. This arrangement determines the characteristics and functions of a virus or a human being cell. DNA is part of every living cell.

Multiple species have biological homology, and evolutionary relationships between them and how they relate to a common ancestor depends on analysis regions of similarity of their DNA. Investigating these similarities requires finding possible alignments in DNA or RNA sequences of these species. MSA helps as the solution and to understand the similarities.

In computational biology, a string represents these sequences with specific characters, for DNA : A, C, G, T. There are additional characters used for calculations representing A or C, G or C, or T. They are ambiguous characters. Table 2.4 shows these relationships.

IUPAC	Code Meaning
A	A
C	C
G	G
T/U	T
M	A or C
R	A or G
W	A or T
S	C or G
Y	C or T
K	G or T
V	A or C or G
H	A or C or T
D	A or G or T
B	C or G or T
N	G or A or T or C

Similarly, Understanding protein sequences in terms of their biological function and similarity using the same approach. A difference is that protein representations have a maximum of 20 unambiguous symbols (Y, R, N, D, C, Q, E, G, I, H, L, K, M, F, P, S, T, W, A, V). A couple of other symbols exist for ambiguity, likewise in DNA,

RNA. However, it only adds to the understating of this work that they exist instead of knowing their tables as done for DNA sequences. Beyond this point, consider DNA and RNA as the same for this work; their differentiation will not affect the final result. This section discusses the background of how to perform these alignments.

2.4.1 Pairwise alignments

The first and more digestible step to understanding sequence alignments is pairwise alignment. Known algorithms in bioinformatics are Needle-Wunsch [17] and Smith-Water [18], commonly known as global and local alignment, respectively. Both of these algorithms use dynamic programming, thus breaking the problem into smaller problems :

1. Initialization
2. Matrix fill (Assigning Scores)
3. Traceback (Generate alignment based on the scores)

The algorithms use a similar approach, becoming possible to understand both algorithms by understanding one and then coming back to the differences. In figure 2.18, it is possible to see the step-by-step. For now, we should understand Needleman-Wunsch [17] algorithm and later comment on what makes Smith-Waterman [18] algorithm different.

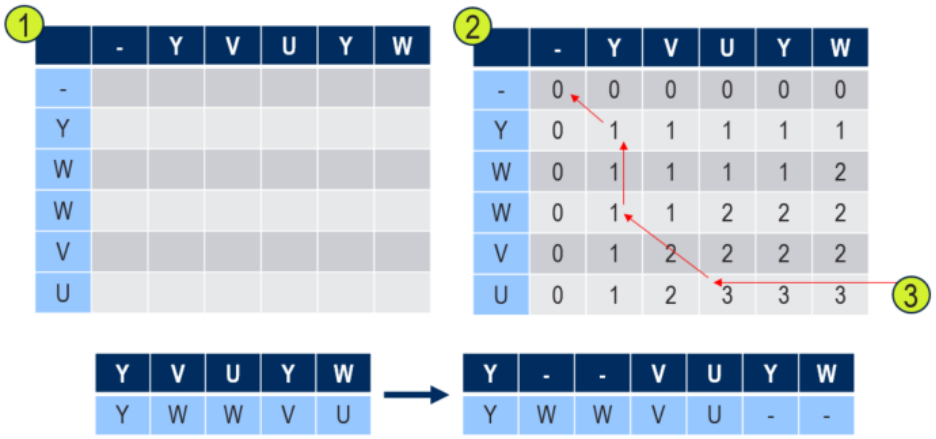


Figure 2.18: Pairwise alignment : 3 steps example

In the first step, a matrix $C_{(N+1)(M+1)}$ is initialized based on the sequences that will be aligned where N and M are the lengths of these sequences. Following that, this matrix is filled up beginning from the topmost left position C_{00} and using a scoring function. The next sections will discuss them. Once every position has been

filled up with its respective score, the next action is traceback. It starts at the highest position possible in $C_{(N+1)(M+1)}$; the indexes are "walked" back until the position C_{00}. The direction of the movement should be to the highest neighboring value. If the movement is diagonal, the characters of that position become part of the resulting strings. If the movement is horizontal or vertical, the resulting strings append a gap('-') and the character that changes with the movement. Note that there are three possibilities :

- Gap: represented by '-' indicating that there is no possible match or mismatch in that point.
- Match: Two identical nucleotides are in that region.
- Mismatch: Two different nucleotides are in the same region based on a scoring system.

The final alignment depends on which algorithm and scoring system it uses. The scoring is done in the second step to fill up the matrix. If the row or column is 0, this is automatically filled with a score of 0. Key aspects of these scoring systems are listed below:

- Gap opening penalty: Score assigned to minimize gap opening possibility.
- Gap extension penalty: Once a gap exists, this score penalizes an extension of gaps in a sequence.
- scoring Matrices: These are matrices that establish relationships and assign similarity scores to different nucleotides based on natural probabilistic observations such as [19] and [20].

A representation of the items discussed above is below. The function that takes a_i and b_j as input returns a score based on the scoring matrix for the Proteins, and DNA has some similarly defined score matrix. W represents Gap opening and extension penalty.

$$\begin{aligned} C_{ij} = MAX(& \\ & C_{(i-1),(\,j-1)} + f(a_i, b_j), \\ & C_{i,(j-1)+w)}, \\ & C_{(i-1),j} + w)) \end{aligned} \tag{2.1}$$

2.4.2 Multiple sequence alignment

MSA is the alignment of 3 or more nucleotide sequences. There are additional alignment methods when MSA, this work makes use of two leading software's methods

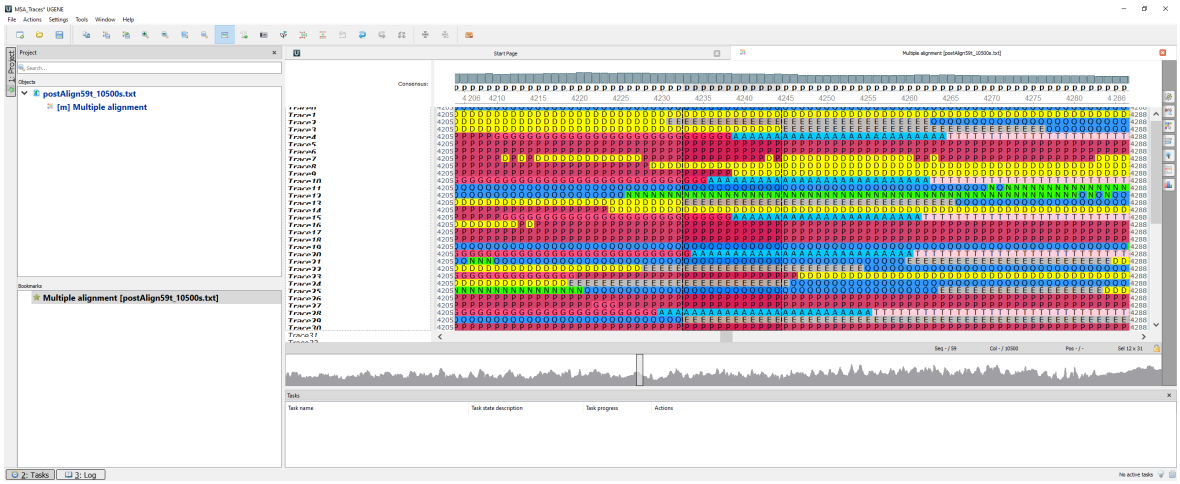


Figure 2.19: Ugene GUI

([21] and [22]). We explore the methods: Dynamic programming, previously mentioned when discussing over [17] and [18]. Progressive and iterative.

Progressive methods achieve alignment by building trees of similarities created by pairwise alignments. It begins with the most similar pair and evolves to the less related ones. These methods can handle large-scale sets (1000 sequences), becoming perfect for this work as it covers the goal needs. Two of the software that has this method implemented to take place in this work as MAFFT([21]) and ClustalW([22]), some others as T-Coffe [23] will be mentioned, but they can not handle as large sequences (around 150 sequences) or are not as fast as the two first mentioned methods.

Iterative methods are also part of the approach used here, and these improve the progressive alignment by iterative realigning the initial sequence and adding new sequences to the MSA. This method can benefit the Progressive alignment by returning to sequences that the progressive method has considered as done. The software MAFFT([21]) and ClustalW([22]) use the previously mentioned methodology. They have an option for the maximum number of iterations.

2.4.3 Bioinformatics software

This work uses software specific to bioinformatics to visualize and perform alignments; this section discusses some of these. A handy tool to visually analyze the DNA / RNA or protein sequences is the UGENE([24]). This software can represent these sequences in an excellent Graphical User Interface (GUI) where it can see different colors for every nucleotide. It has many plugins to call different MSA methods such as MAFFT([21]), ClustalW([22]), T-Coffe([23]), and others.

Given the length of the sequences that this work analyzes, the main software

used for MSA is MAFFT([21]) and ClustalW([22]). The first offers a downloadable executable interface that can be downloaded and used by any programming language by performing a system call, drastically improving the code's usability for this work. ClustalW is also an option for the large sequence created here, but it was most efficient to use it by uploading the sequence to their website and executing it on available clusters. MAFFT also offers the same option. Although these options are suitable for testing large sequence alignment, they are more challenging when measuring method performance.

Chapter 3

Related Work

This chapter discusses the work related to this project. The main topics are SCA and MSA/sequences alignment. An extensive understanding of bioinformatics is not essential, but the idea of MSA's and their final goal. However, that is not the case with SCA. A deeper understanding of what they represent in terms of crypto attacks and why alignments are essential is crucial to the understating of this work. The section 3.1 will discuss SCA, including sub-topics such as alignments and countermeasures. The following section will discuss material related to MSA and its available tools.

3.1 Side channel analysis

The main goal of this work is to provide a method to make side channel analysis possible in countermeasures presence. The work of [5] is groundbreaking to Side channel analysis, and this section discusses it. This work is one of the most important works in this field, and it exposes cryptography devices' vulnerabilities using their power consumption. It becomes possible to extract the secret keys used in cryptography operations. This discovery has helped developers to evaluate and protect their systems against such an attack. A further step from Kocher's work is the CPA [7], which further develops DPA [5] approach by using leakage models and hamming weights [7], the chapter 2 explains this work. In comparison, CPA [7] achieves with a smaller power trace data set what DPA [5] achieves. However, these two methods rely on multiple power trace alignment when the device performs the same operations to expose data-dependent operations leakages. Alignments are also an important theme for SCA, and there are multiple experiments on how to perform them and protect systems against this possibility. The two options investigated in comparison to this work are: Static [4] and Elastic [9] alignments.

Mangrad [4] introduced the static alignment approach for aligning power traces.

It is suitable for aligning traces or sub-regions that have not been affected by countermeasures. According to Mangard [4]: ”The alignment of power traces is usually done based on pattern matching. The alignment technique selects part of the first power trace as a pattern. Subsequently, the attacker tries to find this pattern in all other power traces.”. The benefits of trace alignment make it an important step to be done during cryptography systems attacks. It is usually done in two steps according to [4], finding a pattern that occurs in the first trace and finding the same pattern in the subsequent traces. According to the same work, define a couple of features and how to choose between them. These are Uniqueness, Data Dependency, Length, and distance to the attacked intermediate result. The author explains that length does not mean that the most prolonged pattern is the best, and it requires a proper investigation to choose a pattern. Furthermore, suppose the time distance between the alignment point of the chosen pattern and the attacker's target is considerable. In that case, the region of interest likely will not be aligned as countermeasures might have acted and altered the traces. This work also discusses pattern-matching techniques based on least squares and correlation. In terms of performance, the static alignment is the baseline for comparison. This alignment method is typically the best option for alignments where no countermeasures exist.

However, it is only sometimes the case that static alignment is the best choice for the power traces, especially when countermeasures are in place. It is possible to find in the literature exciting solutions to this problem. The Elastic alignment [9] is one of them. The elastic alignment uses methodologies first developed for speech recognition: Dynamic Time Warping (DTW) [25]. The problems in speech recognition can be pretty similar to the ones faced when comparing power traces with countermeasures. That is why using the approach presented in [25] works for this. The [9] explains that words can be spoken with variances in timing and identifying and comparing them. It is most certainly a non-trivial problem. Spoken words can not simply be compared to pre-recorded samples in a sample-per-sample approach.DTW [25] is based on dynamic programming, and it measures the distance between two utterances by elastically warping them in time according to [9]. Given the computation complexity of DTW [25], the work of elastic alignment goes a step forward by improving performance by using fastDTW, thus optimizing its results in terms of execution performance when it comes to power traces alignment. See an example of this alignment in Fig. 3.1 where 1 represents the source traces and two the post elastic aligned traces. Results compared to the previous method (Static alignment) are also presented. It shows CPA success rate of the alignment on traces with countermeasures to prevent SCA. Compared to a fixed length clock cycles trace set, static alignment uses 1400 traces to obtain a success rate of around 50%. The elastic alignment achieves a success rate close to 100% for the same trace set of

around 270 traces. Elastic alignment is also proven to be suited for unstable clock cycles being relatively unaffected by them.

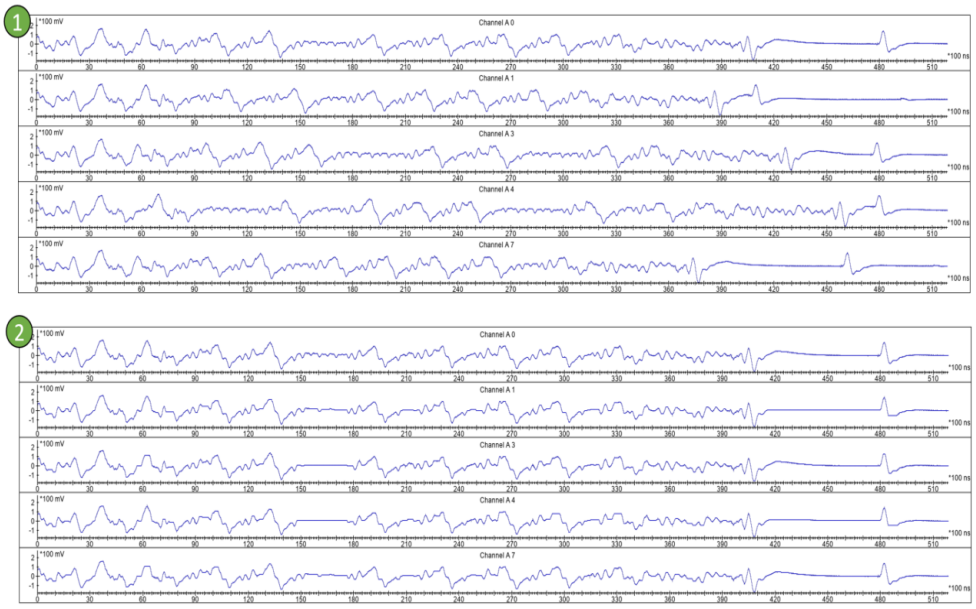


Figure 3.1: 5 Traces elastic alignment example

Rapid Alignment Method (RAM) [26] is an alignment method also inspired by another field of expertise. It derives from image processing algorithms. The U-SURF [27] inspires the algorithm. This approach uses a reference picture to recognize multiple pictures of the same reference image. A couple of techniques presented by [27] are used to achieve its goals which are high impact in terms of execution performance by the use of block wavelets, having as the main advantage the running time of O(1), achieving the 20% faster execution performance when compared to elastic alignment.

Once again, to find vulnerabilities in implementations with countermeasures,SW-DPA [28] proposes a solution to find the keys of the cryptography operations where random process interrupts are in place as countermeasures. SW-DPA [28] approaches the problem assuming that clock cycles have fixed Lengths and their averaging number is specified based on the number of random process interrupts. Based on these two assumptions, it integrates the leakage that was distributed over a few clock cycles and makes the approach proposed by Paul Kocher [5] possible once again. Comparing the methods discussed here, assuming fixed-length clock cycles, static alignment uses 1400 traces to obtain a success rate of around 50%. The elastic alignment achieves a success rate close to 100% for the same trace set of around 270 traces, and SW-DPA achieves the same success rate with 160 traces.

The table below is the comparison of the results shared by [26]. In the table below, he mentions that although static and SW-DPA alignments are faster, they do

not achieve DPA or CPA with the alignments. In terms of DPA success rateRAM outperforms elastic alignment by order of magnitude.

Method	Run Time	Time Per Trace
Static Alignment	12 minutes	1.44
SW-DPA	18 minutes	2.16 ms
RAM	76 Minutes	9.1 ms
Elastic alignment	3115 minutes	373.8 ms

3.2 Time series and Bio informatics

In order to be successful, this work needs to align time series, for this specific case: power traces. The alignment requires this work to bridge the knowledge between two fields that, at first glance, are not related, hardware security and bioinformatics. This approach will depend on transforming the power traces to strings of characters that are, in reality, representing nucleotides that, when grouped in a sequence, either create a DNA or amino acids that create proteins. This section discussed the literature with a similar need for conversions to perform time series processing and investigate information.

The first related work we present [29]. It presents a random delay identification method in power measurements by creating strings based on hamming weights of predetermined opcodes and using string matching algorithms to find patterns and generate a final alignment. [29] achieves success by understanding the operations performed by their devices and reducing the total power consumption of one operation to one sample. Thus the resulting strings are a sequence of opcodes represented by one character. This approach allows them to use a generalized Bayer-Moore-Horspool algorithm [30] to detect the random delay and later find correct alignments. A work investigated as an intermediate conversion step is Symbolic Aggregate Approximation (SaX) [31]. SaX proposed a unique method to transform time series into a symbolic representation, and it aims for similar goals of other methods that this work presents. SaX uses Piecewise Aggregate Approximation (PAA) [32] as a dimensionality reduction step. PAA reduces the dimensionality by simply reducing a time series from n dimensions to ”w” dimensions. Thus, in the new time series, each of the ”w” samples represents the mean of n samples. The formula is below:

$$C'_i = \frac{n}{w} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} C_j \tag{3.1}$$

After the dimensionality reduction, SaX has as the next step discretization. This discretization starts by normalizing the time series after performing PAA. This step produces equiprobable symbols as the work states that normalized time series have Gaussian distribution. See fig 3.2 and 3.3.

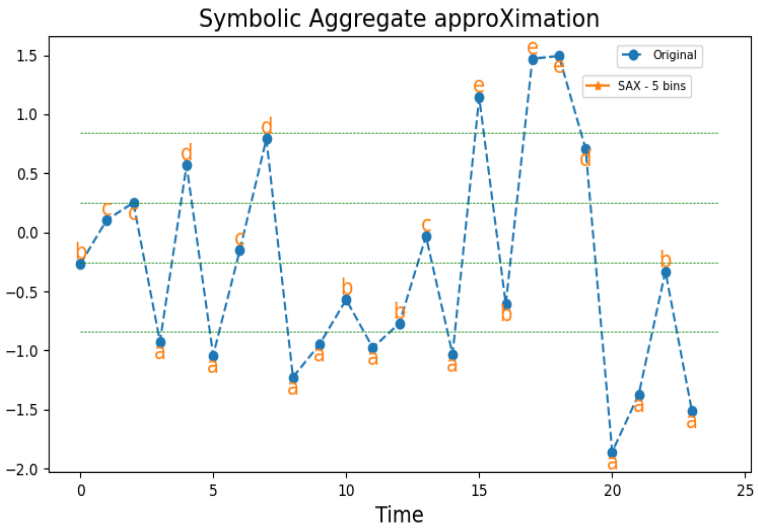


Figure 3.2: SaX transformation

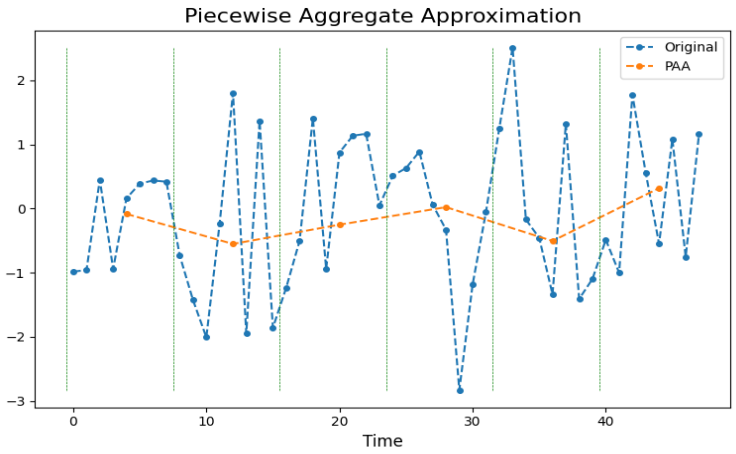


Figure 3.3: PAA transformation

Identifying previously known and unknown patterns using specific algorithms is vital for reducing dimensionality and transforming a trace to a character sequence in a more meaningful approach to represent a nucleotide sequence. Some methods for time series classifications are beneficial for this research. [33] defines a methodology called DiscMotif to find motifs, defined as the frequently occurring patterns in a time series. They aim to find the K most significant motifs in a time series. It also uses the transformation of time series into symbols by using SaX [31] as an intermediate step.

3.3 Discussion and Conclusion

This related work section has shared and discussed ideas from three different fields of study that this work will investigate to propose a novel solution for SCA. This work closes a gap between SCA and Bioinformatics investigation options in digital signal processing. Bioinformatics evolved substantially in recent years due to investigations during Covid [34] pandemic that demanded high performance from this field. Other fields that rely on performing time series alignments can benefit from that.

This chapter presented SCA existing alignments([9], [4] & [35]) comparing their results and achievements, these alignments use an all-to-one approach, and this work proposes an all-to-all approach. It also presented the ideas from Bioinformatics, and how this field uses its alignments to provide essential information to scientist performing their research([19], [20], [17] & [18]), to the best knowledge of this author, there is not yet a work that targets SCA with bioinformatics alignments(MSA). The last and crucial related work investigated is the transformation of time series, which is essential to this work as time series need conversion to nucleotide sequences ([33], [31] & [36]).

Chapter 4

Methodology

This section discusses the methods used to achieve this work goal and the design choices made throughout development. The first section will discuss the implementation overview. Further, the chapter discusses in more depth the implementation details. In the case of multiple options available for a single step, subsections explain them.

4.1 Power traces Bioalignment: Overview

Although not similar at first glance, power traces, and nucleotide sequences have much in common when compared under the appropriate circumstances. A nucleotide sequence defines the characteristics of cells or viruses and biological information, and power traces describe a device's behavior during an operation and can be used to extract information. When countermeasures in power traces, these can be compared to mutations when looking at DNA or proteins. MSA can identify mutations in nucleotide sequences using an all-to-all alignment. Analyzing diseases requires these techniques, for example [34].

The idea MSA inspires this work to deliver an alignment of power traces, translating and adapting the knowledge available in bioinformatics as a solution. The overview of the algorithm is below. All steps have corresponding numbers in Fig. 4.2. See also these steps in a detailed flow chart in fig. 4.1.

- **Select power trace (1-2):** the trace set is defined, as well as its sample range that will be aligned.
- **Conversion from power trace samples to nucleotide (3):** that is the conversion from the power trace into the DNA or protein representation. A couple of options are available to achieve the conversion. This chapter and the results chapter discuss them.

- **Multiple Sequence Alignment (4-5):** Once the nucleotide sequences representing the trace sets are ready, existing methods in the bioinformatics field perform the alignment. It is also important to discuss the options available for bio alignments. These different methods and their parameters can greatly impact the final result.

- **Conversion from MSA to power traces (6):** When alignments are ready, Another step is required: transforming nucleotides into their original values. When sequences are re-transformed into bytes representing initial numbers.

- **Generating new power trace files:** the traces can now be output as files that contains the MSA aligned power traces data.

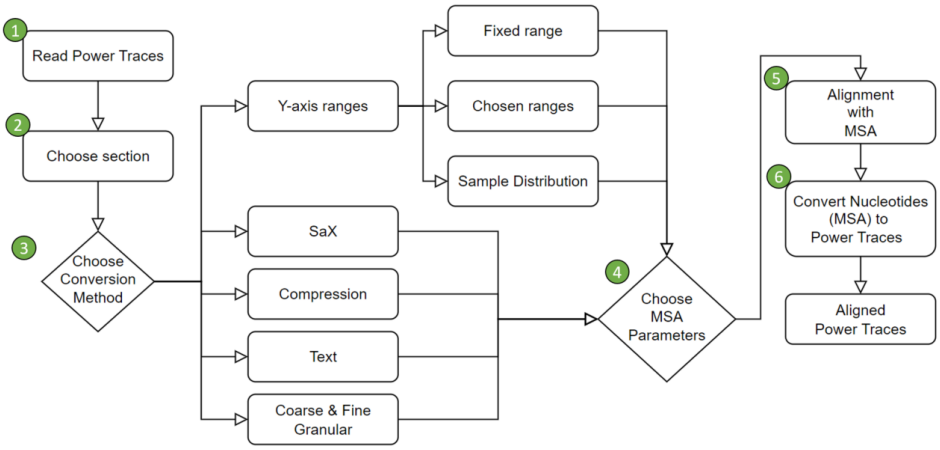


Figure 4.1: SCA with MSA : Flowchart

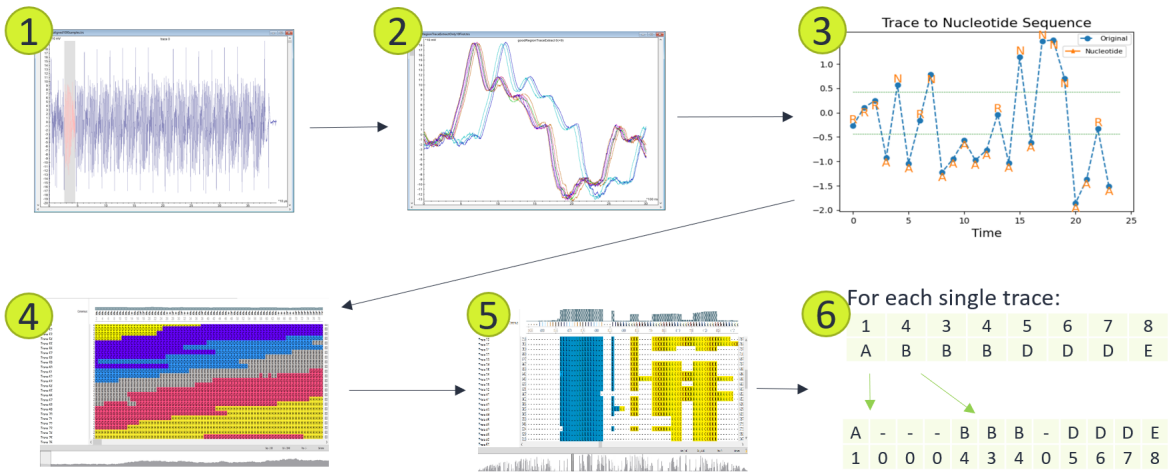


Figure 4.2: SCA with MSA steps

4.2 Power Trace to Nucleotide Conversion

The challenge on this topic is how to represent the power trace samples into the symbols that will be input for the MSA. Every sample of any trace in this work is a byte, therefor a value varying from 0 to 255 will be converted into a maximum of 20 symbols (maximum of 7 symbols if DNA).

A wide range of approaches was experimented with to evaluate the impact of the conversion step into the algorithm. From transforming a single sample to a single sample, using a fixed range of sample value references per symbol to developing a method that would take the average sample value distribution of all traces and use chosen ranges by the user. Given that the amount of traces and samples can exponentially impact the algorithmic complexity of the alignments, this work includes the development of other methods, such as compression of symbols and other attempts by using what is available in the literature for time series processing that could be applying to this work, such as SaX [31] and PAA [32].

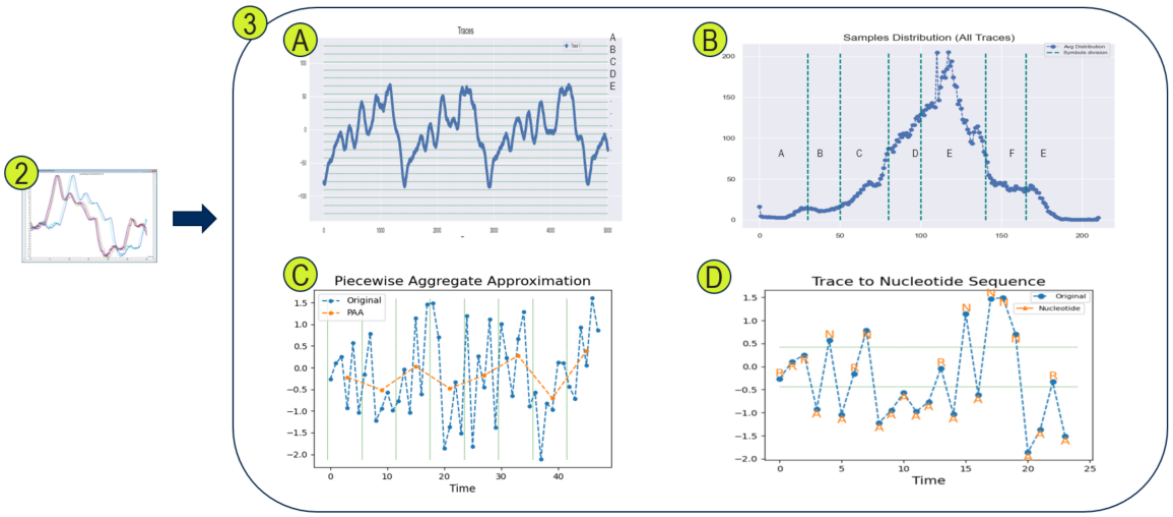


Figure 4.3: Trace to nucleotide conversion examples

This work experiments with the conversion options with their variations to achieve different results. The most used of them is the Y-axis division in equal-length regions. One method divides the Y-axis range into equal parts based on the alphabet length of nucleotide characters. For that, the first step is finding all traces' maximum and minimum values. For example, we want to convert a trace into ten nucleotides, with a maximum of 200 and a minimum of 10. The number 190 is divided by 10. The range from 10 to 20 is the first alphabet letter, 20 to 30 will be the second, and so on (step 3.A from Fig.4.3). The minimum and maximum values are not defined per trace but for the entire trace set. Another option available is that ranges could also vary by having symbols that contain a more comprehensive range of samples than

others.

Similarly, range definitions come from sample distribution. This approach uses the average sample distribution of all traces, and based on ranges defined by an input, it converts to symbols as in step 3.B from Fig.4.2. Given the algorithmic complexity of the MSA implementation and the number of samples most of the traces contain, implementing alternatives for sample compression was also a concern in the development of this work. In 3.C from Fig.4.2, PAA and this work's compression idea are used as an option for compression as this algorithm compresses multiple samples into an average sample for every compression group size defined, the goal here was to attempt performing MSA on smaller sequences maximizing performance. Compression will be discussed further in another section of this chapter4.2.3. SaX [31] was also included as a conversion method.

4.2.1 Y-axis conversion

This work uses several conversion methods focused on the Y-Axis division/conversion into ranges. The first to be discussed will be the most straightforward of them. Converting in the Y-Axis by using ranges equally divided based on the number of desired regions. To perform this transformation, let us discuss first some steps in the process:

1. Define alphabet
2. Define the minimal and maximum value of the trace set
3. Define range per symbol
4. Create a new sequence of symbols

The algorithm for this implementation starts by defining an alphabet. For proteins, the user can choose the size of the alphabet to a maximum size of 20. For the DNA option, this work will always use seven symbols: 4 DNA main symbols and three ambiguous symbols, representing the areas between the main symbols of the alphabet, to maximize possibilities of mismatches.

The second step is to find a minimum and maximum finding function on all the traces of the trace set and keep smaller minimum and higher maximum, allowing us to define the ranges for the symbols of the alphabet. For example, assume we are using DNA and the minimum is -20 and the maximum 50. The range will be then 70 divided by 7. The symbols will then represent: ”A” = (-20) to (-10), ”M” $>$ (-10) to 0, ..., ”T” = 40 to 50. From step 3, step 4 creates new arrays that represent the trace set with sequences of the alphabet chosen, making the use of the MSA possible.

An example is in fig. 4.5, where two different traces in the traces sets are. The minimum and maximum of the trace set are -89 and 112, respectively. The calculation done is ((112+89)/7). The result represents the large zones marked by the green horizontal lines, starting at -89 and finishing at 112. The symbols on the left represent which character in the DNA alphabet represents the sample will in the conversion step.

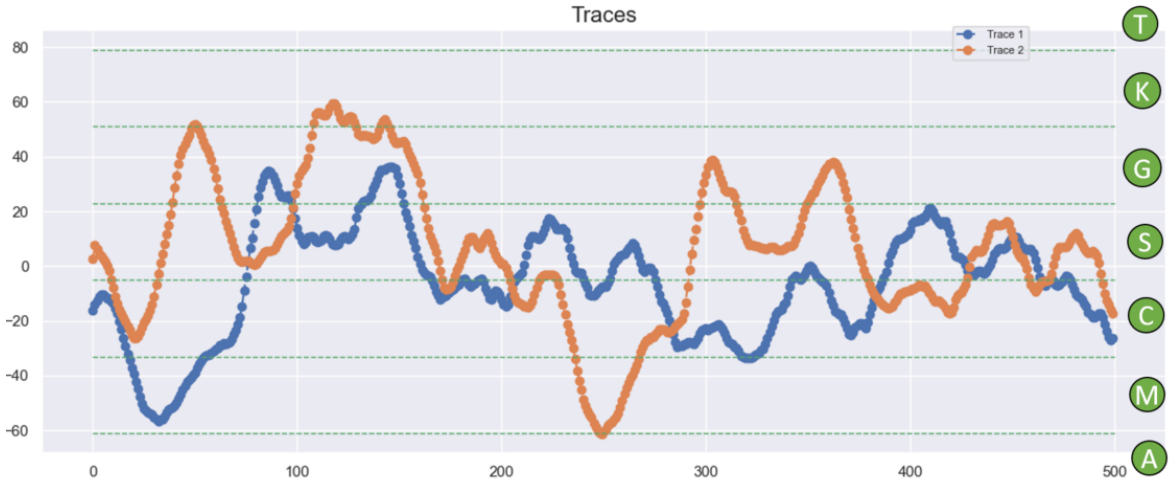


Figure 4.4: Traces to DNA example

Although in terms of Y-Axis, the approach explained until now was the most used, several other options are available in the implementation. See below the options summarized :

1. Conversion to Proteins (Max. 20 symbols)
2. Conversion to DNA
3. Conversion to text/ASCII
4. Variable ranges per symbol
5. SaX

In the items from 1 to 3, although the result varies from each other, the intermediate steps are similar. They use a different approach of transforming the sample values of the traces to symbols, where the most significant impact/difference in the result will happen by the alignment step, explained later in this chapter. Item 3 uses the 128 chars presented in ASCII to represent 128 zones of our traces. In the 4th item, the user can provide the alignment function with an array of how he wants the conversion ranges to be. So instead of using fixed ranges based on the min/max and division by alphabet length, the user can provide an array containing at which

point the range divisions should be. Take, for example, the alphabet ”A”, ”B”, ”C” and ”D” and the array [20,50,60]. Any value below or equal to 20 will be ”A”, ”B” between 21 and 50.

The most diverging approach is using SaX [31], an algorithm widely used in the literature when data mining on time series. This work adapts it to this purpose and compares it against the methods developed here.

4.2.2 Average Samples Distribution conversion

The idea comes from the curiosity that extreme ends of the traces could or not use more or less granularity, and this method exists to test that. Instead of looking into the Y-Axis and defining regions for the conversion, this method takes the average distribution on samples of the trace set. It uses the idea of variable ranges to perform the conversion. The user then can input an array that defines the ranges, just as the variable ranges per symbol option in the previous subsection. From there, create nucleotide sequences that represent the traces.

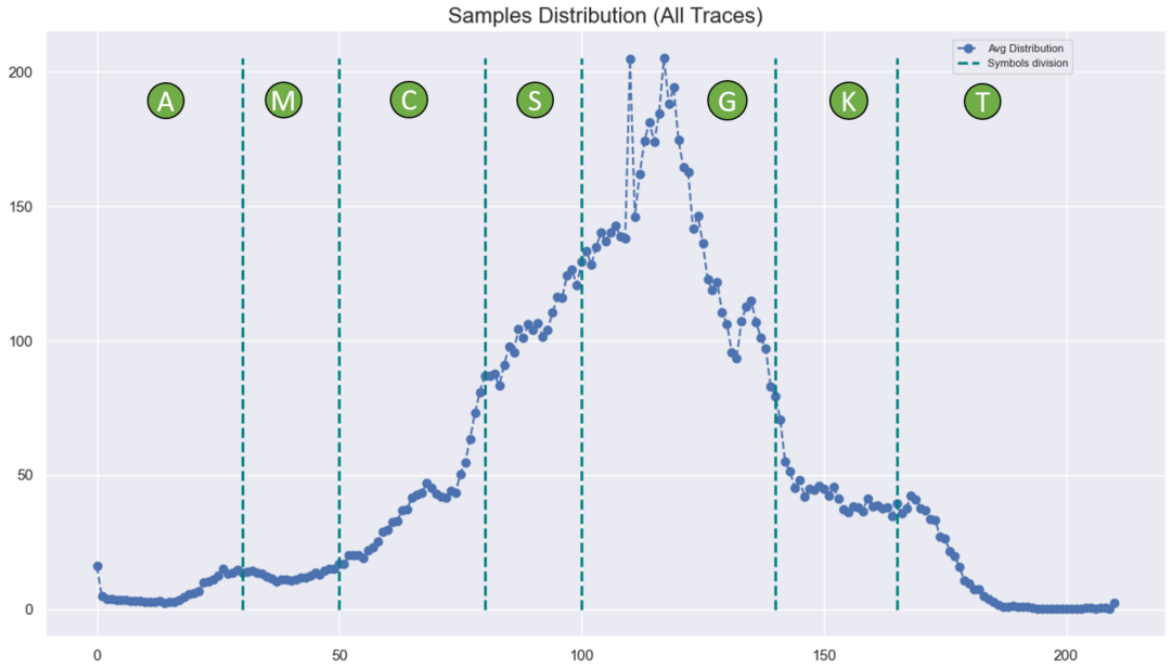


Figure 4.5: Traces to DNA example based on average sample distribution

4.2.3 Compression

Some implementations test the impact of the reduction in execution time and if the alignments are possible. For MSA , the number of sequences and their length has an exponential impact. Apart from this work's methods, the algorithm includes the PAA [36] algorithm.

1. Compression with Pre-defined number
2. Compression by Region
3. PAA

PAA is also included as an intermediate step for SaX. It averages the samples in a specific region defined as the window size. This value can then converts into a symbol using one of the methods previously discussed in the subsections of this section.

Another two options were created based on the idea, but the focus lies in compressing the symbols representing trace samples. Item 1 compresses equal symbols based on a maximum compression number. For example, assume compression of 3 and the sequence "AAACCCCCCDDD". The resulting sequence will be "ACCD". Item 2 behaves similarly, but it always maximizes compression. For the same initial sequence, the result would be "ACD". For all compression approaches, an array of compression indexes is available for decompression. Every index of this array has the number of symbols it represents. As the item 2 result, the array would be [3,6,3] for "ACD". They represent the original sequence, which had 3 "A", followed by 6 "C" and 3 "D".

4.3 Exploring Scoring Matrices

An essential feature of the MSA is that it takes into account gap opening and extension penalties and also uses scoring matrices ([19], [20]). That is because although some amino acids in a sequence might not be the same, they can assume similar functions in a sequence and could be considered a mismatch. As mismatch, understand characters/symbols that, although different, are considered similar, given a score. A sequence of characters alignment techniques considers whether the characters at the same position are identical. The use of scoring matrices for similarities makes this algorithm translatable to the goals of this work. As we translate and apply this approach to power traces, it is vital to understand how it can benefit from these ideas.

Figure 4.6: Power trace to nucleotide zone

Figure 4.6 illustrate the symbol regions of the traces used to produce a final alignment that complies with our final goal. These traces are aligned and provide visual assistance in understanding the design choice. Every green line in the figure represents the range in which a sample transforms into a specific character. Both traces represent the same operation simultaneously, but there is slightly lower power consumption at trace 2 in region 1, which is data-dependent. Analyzing possible conversion to characters/nucleotide sequences, we could expect trace 1 in blue: ”EDCE” and trace 2: ”EDDE”. It would be much longer with the real number of samples, but let us accept this short sequence for better understanding. The resulting alignment could be: ”ED-CE” and ”EDD-E”. In this case, those gaps should not be there. Avoiding alignment results like the one described is key. This work uses scoring matrices for proteins and ambiguous representations if performing DNA alignments. See the alignment in fig. 4.7.

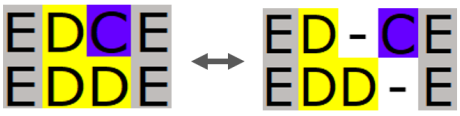


Figure 4.7: Two sequence alignment example

In the case of power traces, it is desirable that both in regions like 1 and 2. ”A” is a match with ”B”, sometimes even ”C”. However, ”A” should never be aligned with a ”G” or ”H”. The problem solution uses a protein scoring matrix that provides some ambiguity of symbols. Every region’s symbol is similar to its neighboring regions. The scoring matrix defines this to maximize the possibility of mismatches and minimize opening gaps in regions shown in fig. 4.6. This work uses Multiple Alignment Fast Fourier Transform (MAFFT) for alignments and the Blosum 62 [20] as a scoring matrix. DNA alignments use a relatable idea. However, without a scoring matrix. The main symbols are ”A”, ”T”, ”G” and ”C”. For the regions between these char-

acters, this work uses ambiguous representations. For example, ”M” is understood either as ”A” or ”C” meaning that ”ACC” and ”AMC” would be understood as the same sequence. Chapter 2 defines the relations.

4.4 MSA

Different options available for bioalignments in this work are: ([37], [38], [39], [40], [41]). The trace set length and software availability determined the method for developing these ideas. MAFFT [21] was widely used as it can run on your machine by using a system call during the execution of this solution. Other solutions, such as ClustalW [22] and T-Coffee [23] were also tested. However, they only accepted up to 150 sequences or would need to be run online on their servers, leaving MAFFT as the most suitable solution.

4.5 Power Traces from MSA Conversion

There are different conversion options implemented during the execution of this work. The idea was to have multiple options when investigating the results. Once MAFFT performs MSA, the sequence needs to become a trace set again. The alignment done by MSA approach does not remove any information, meaning that only gaps are new to the existing sequences. If needed for evaluation purposes, this operation is reversible: finding the original nucleotide sequence can be done by removing the gaps from the result sequence. The challenge in this transformation is interpreting the added gaps as samples since these were not part of the originally sampled traces. Note that there is no lost information for any conversion methods discussed in this section.

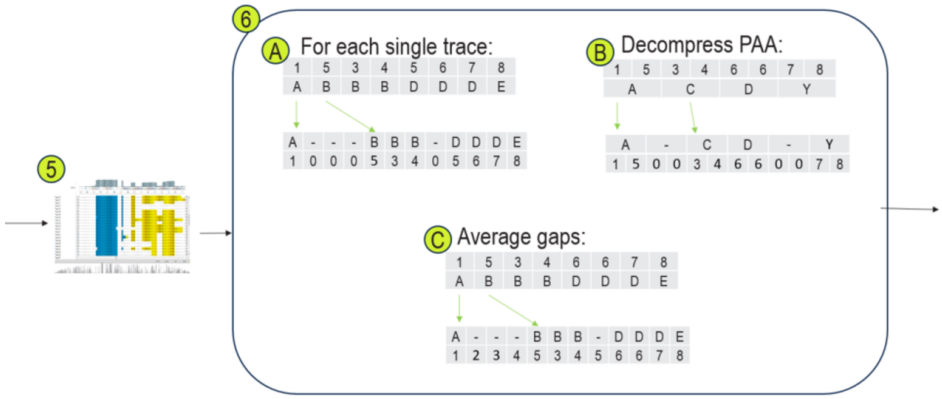


Figure 4.8: MSA to power trace conversion

There are three options available for conversion from aligned sequences to power traces again:

1. Gaps become zeros

2. Gaps are interpolated using the numbers between them

3. Gaps are multiplied based on compression factor, and then item 1 applies

For the conversion method in item 1, the gaps become zeros. The results are achieved by running through the aligned array of characters and creating a new array/trace. If a character that is not ”-” is read, the value is inserted into the new array. For a ”-”, insert a 0. Fig. 4.8 by 6.A represents the process. Item 2 is represented in the same figure by 6.B is a previous step. If used, a character represented an average of a chosen number of samples. Thus for every trace transformed to a nucleotide sequence, a new array with the compression per character reference is stored. A reference array per character is stored if different amounts per character exist. Otherwise, only one integer holds the count. In the figure, the compression size is two. Thus ”A” represents 1 and 5, and a gap converts the same length number 2. For item 2, the gaps are also converted back to bytes, following the idea in item 1.

Item 3 was developed for the case that adding zeros impacts the final result(CPA). The idea here is that the gaps become an interpolation of the numbers between them. In the 6.C example, instead of adding zero between 1 and 5, these numbers become 2,3 and 4. This calculation is done by adding the two edges (1 + 5 = 6) and dividing by the number of gaps in that space (6 / 3 = 1). The result of each gap is the division result added to the previous index number for each gap (1+1 = 2, 2 + 1 = 3, ...). The division is always rounded to the closest integer, as a byte must represent the results. The subsections of this chapter explain all these methods.

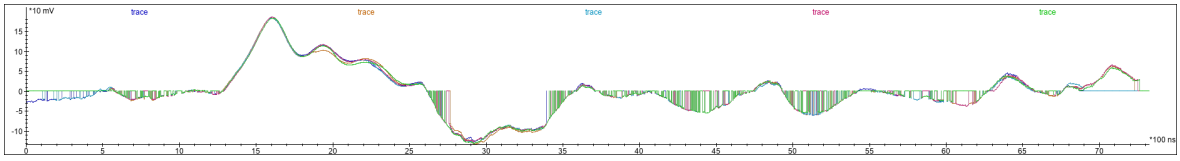


Figure 4.9: Gaps as zeros

Fig. 4.9 and 4.10 present the approaches described after an alignment is processed and reconverted into power traces. Item 3 is visually the same as in fig. 4.9, thus it will not be demonstraded.

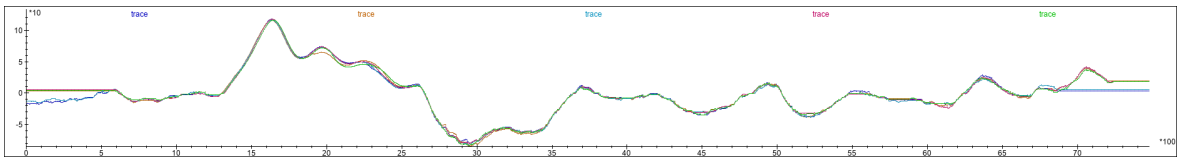


Figure 4.10: Gaps removed by averaging samples

4.5.1 Y-axis and Sample distribution

After the MSA methods perform the alignments, the translation back to power traces needs to happen, as mentioned before. Gaps may separate characters/symbols ”-”. Remove them, and the sequence is again original.

The behavior here is the same for all types of transformations mentioned in the section about conversions from power traces to symbols, apart from the ones that used compression. The algorithm will follow the steps below:

1. Initialize current index as 0 and symbols sequence index as 0
2. Read current sequence index
3. If ”Character”, append sample trace of the current index, and increment the current index.
4. If gap(”-”), append zero to sample index,
5. Increment symbol sequence index.
6. Go to 3

This implementation is independent of the conversion method from power traces to symbols used prior to the MSA, as we need to bring back the samples, mainly because in all alignments, no data is removed or added besides gaps.

Gaps removal is an extra step done by a function that verifies what is and what is not a gap in the power trace based on the MSA and they result in the power traces showed in fig.4.9 and 4.10.

4.5.2 Decompression

The post-alignment of compressed MSA using the methods presented in this work has a special step. It is, in fact, equal to the three of them. An array that determines the compression is stored and used for this step. It represents the compression for each sequence. See the example in fig. 4.11.

There are a few exceptional cases of decompression. These require a particular approach to guarantee the same length. That is done by padding the final results

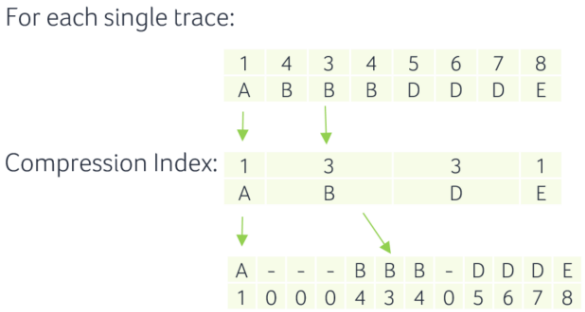


Figure 4.11: Gaps removed by averaging of samples

of the decompression with ”-”(gaps), which will become zeros. Traces are never decompressed by themselves, especially if variable compression lengths are in place; for example, in the same column of an aligned compressed trace set, there are ”A” (150 samples), ”B” (120 samples), and ”-”.

4.6 Consensus

A function called consensus delivers more information over the alignments done with the MSA approach,

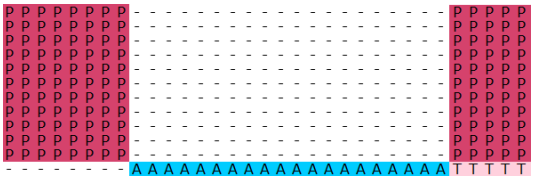


Figure 4.12: Ugene alignment Result

Fig. 4.12 shows the alignment of 12 traces. The consensus would be a count of how many non-gaps sites the traces have per column. For this particular case, the consensus would be 11 for the first eight samples, decrease to 1 where there is only one sequence that has ”A” and ten at the region where the ”A” ends. This information is particularly interesting as extra information to a trace, given that a low consensus can identify regions of countermeasures, as countermeasures would not align with other parts of the traces. The plots in Fig. 4.14 and 4.13 show three out of 10 traces and their consensus. When the consensus is too low, the region identifies a random delay in one or more traces. On the other hand, when it is higher, it is more likely to be a standard operation for all traces.

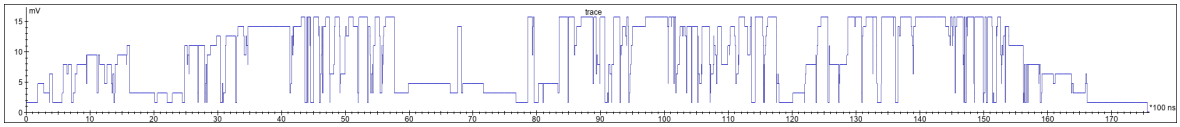


Figure 4.13: Consensus

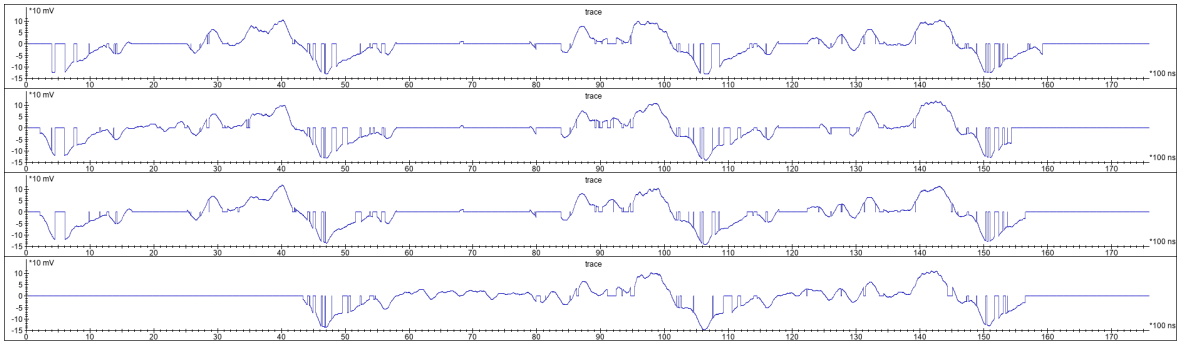


Figure 4.14: 3 Traces with countermeasures alignment

4.7 Static alignment MSA based

This work also developed an alignment method based on the resulting gaps in consensus as an investigation step for this work. The goal was to perform an initial step where an alignment is done similarly to the static alignment. The MSA uses gaps to shift samples to create better alignments, knowing that it is possible to assume we can shift samples in the original trace by the number of gaps added up to that point for every trace. This method takes the following steps:

- Find the longest connected section in the post-aligned trace
- For each trace, count how many gaps were found from that point to the beginning
- Use the gaps count for each trace as a shift value to each original correspondent trace in the trace set

The result is in fig. 4.15 and it is comparable to the existing static alignment.

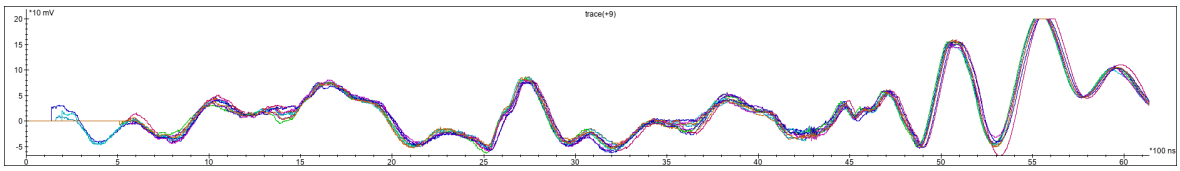


Figure 4.15: Shifting with gaps count

Fig. 4.16 has more details of the intermediate step. The overlapping traces and the longest connected region of all traces are in the green box. From the green box

to the beginning of the trace for each sample, the gaps count indicates how many shift units are applied per trace, as pointed out in the green arrow.

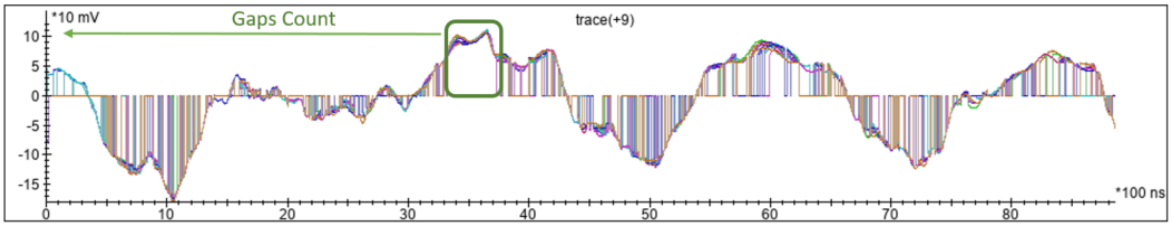


Figure 4.16: Shifting with gaps count: Intermediate step

4.8 Two steps alignment

This section discusses fine and coarse granularity as an option for the alignments. Given the execution complexity of alignment algorithms, the section presents an initial solution, an alignment option using the step coarse and granular approach. The idea is to use a higher level alignment as the first step (Coarse) to align subsequences and only then use a more granular approach to perform alignment between these prior aligned sub-sets. Once these sub-alignments are ready, the final traces can be concatenated to output the resulting complete alignment. Fig. 4.17 represents the approach where the regions ”A” and ”B” are the coarse regions for alignment. In the second step, the alignment technique focuses only on the same region with its similar regions. This approach attempts to improve the time performance as the alignments happen for smaller sequences.

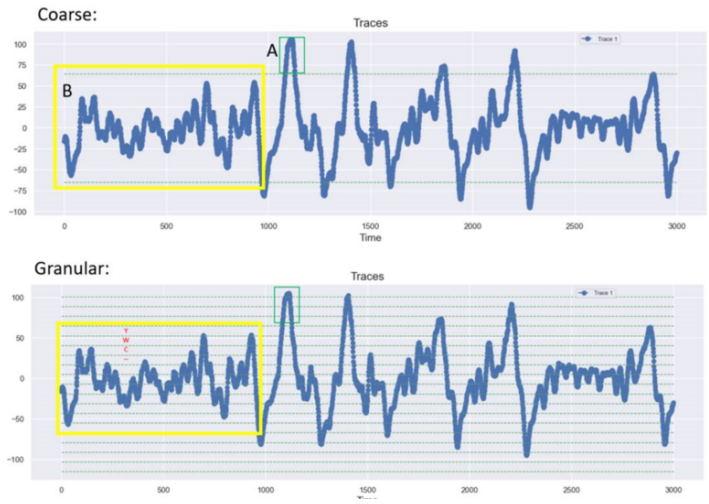


Figure 4.17: Coarse and fine granularity

Chapter 5

Implementation

This section describes implementation details, technologies, and tools to develop and deliver this solution. That includes software to programming languages used in the implementation of this work.

5.1 Riscure Inspector

The software Riscure Inspector [16] has a significant role in this work. It aids three main actions: traces visualization, performing any preprocessing on the traces, creating specific sections of the traces and saving new files, and post-alignment analysis. However, this work implementation does most of the processing. The most important features for post-alignment are known-key analysis and first-order analysis. This work used these processing steps as evaluation methods for this work. The chapter 2 describes them. Training for this tool is available with Riscure. See fig. 5.1, the inspector interface when a trace and its average samples distribution plot.

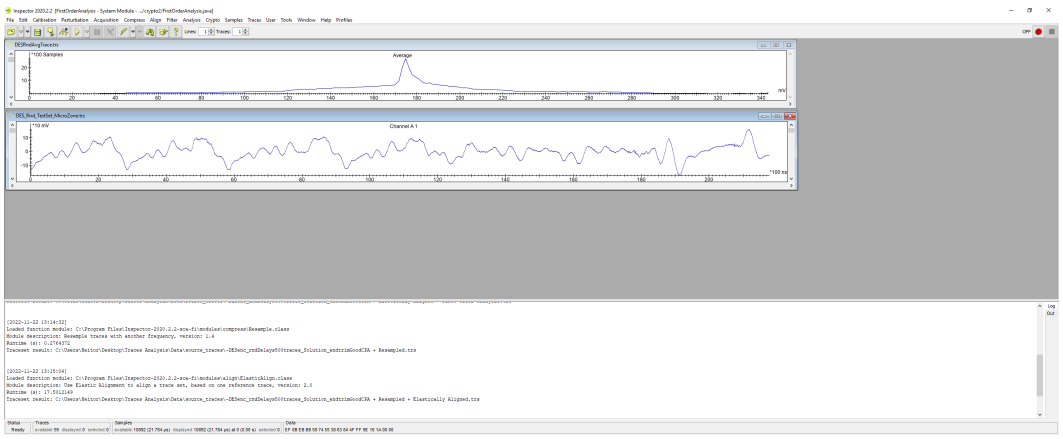


Figure 5.1: Riscure Inspector

5.2 Power Trace Signal processing

Python [42] is the development language for this work's software to process the power traces. This programming language was chosen due to its straightforward syntax and for being relatively fast in performing trace processing. Another practical reason is that Riscure Inspector uses the ".trs" file format for the traces it operates on for this work. A library to process ".trs" files is published [43]. It is well documented and available online for processing the power traces. This work uses both python and the ".trs" files format. Please note that ".trs" files are power traces. That is the format that the Riscure inspector uses to store data. For example, if the module of finding the samples' average distribution of the traces is the choice, that data will be stored as ".trs," and the software developed by this work can process it.

5.3 MSA tools

As mentioned before, a wide range of tools are available for processing the MSA. They are all focused on biological alignments and offer different options based on goals that scientists from that field would want. This work had to investigate what would be the most appropriate for this new method. As per the reasons explained before, the tool chosen is MAFFT. Python uses a system call during the execution of the alignment program (MAFFT). It receives the parameters and returns the final alignment. The recommended version of this tool is available for Linux. This work is implemented in windows but uses a version of embedded Linux (Ubuntu on windows). The installation procedure is described by the MAFFT website [21].

The main goal was to run the alignment methods on multiple large sets and decide to best approaches and software to run on the local machine. Online platforms were also used, although using them was challenging due to the inability to measure execution time, as it would be running in platforms that we could not guarantee would be precisely the same every time. Fig. 5.2 [21] demonstrates the visual of such a platform.

5.4 FASTA Files

The MSA tools use data on specific formats. This work used FASTA [44] as that could be used by MAFFT for the alignment and UGENE [24] for visualization. In this representation, files are in a format where names after ">" represent the names of the sequences and traces in this work approach. See the example below:

. .

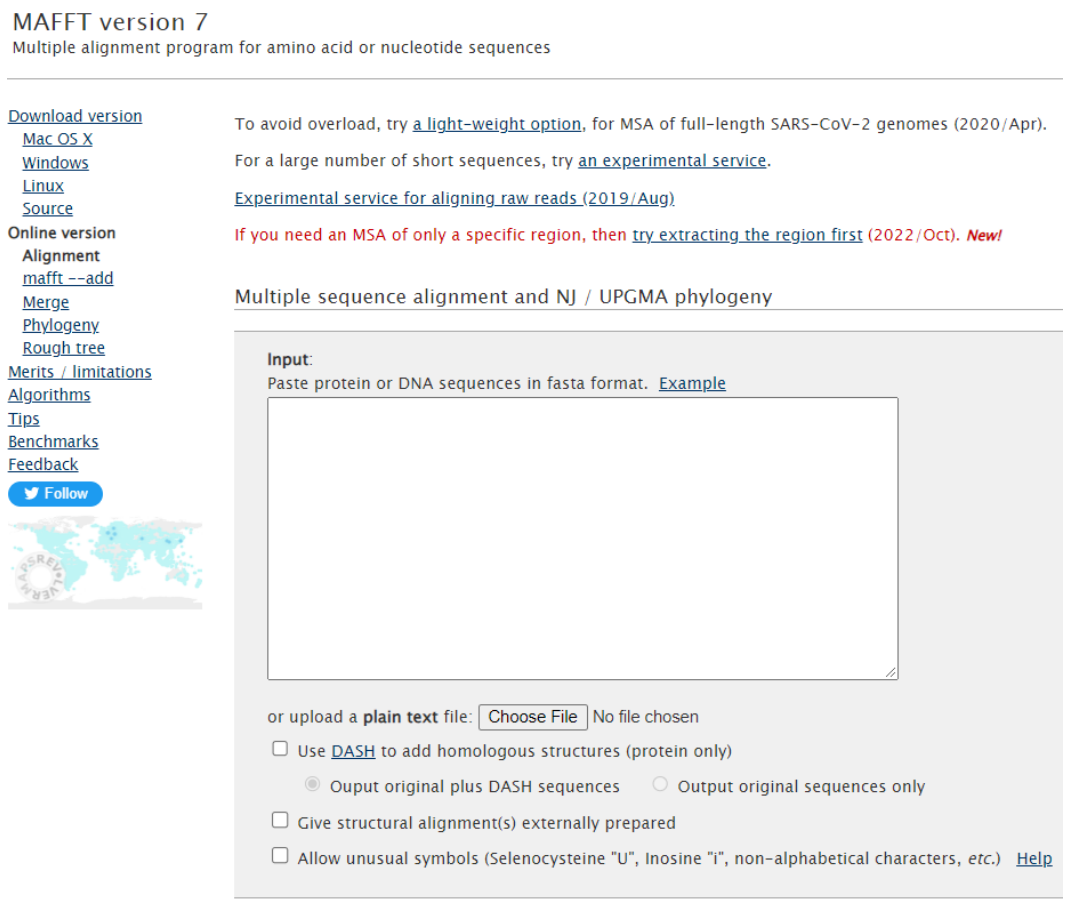


Figure 5.2: Online alignment platform

>TRACE10
MTEITAAMVKELREDCCDLGKAAKKADRLAAE
ABYWWWWYYWWAAASTGAGMMDCKNALSETNG
>TRACE11
. .

This file contains multiple sequences aligned or not aligned. They have, as mentioned before, the name of the sequence after > in the first line. The following lines have characters representing the symbols of that sample (named ”sites” in the genetics field) in the sequence. Every line contains 16 symbols.

5.5 Code development

This project uses Python [42] to develop its functions; as mentioned before, it uses *.trs files for power traces, and Riscure Inspector stores its data using this extension. For this, we use libraries that Riscure distributes for manipulating such files [43]. This work has implemented its code for every other processing step. The source code is

available on the author's github [45] for all other functionalities. It also includes files that explain how to use it for its purpose.

Chapter 6

Results

This chapter discusses the results of the method developed and its variations. The first aspects investigated are the MSA parameters and their impact on the final alignment. The next step is to compare it with existing methods ([4] & [9]). Although the method aims to increase attack effectiveness when countermeasures are in place, we present first traces without countermeasures. It compares its result to static alignment, which would be enough to solve the misalignment. Later sections will demonstrate it in the presence of countermeasures. This chapter presents the performance results at its end.

6.1 Experimental setup

The power traces are power consumption of the pinãta [46] board from the Riscure company. This board has an ARM Cortex-M4F core that operates at 168MHz clock speed. It targets training usage on SCA, DPA, and Differential Fault Analysis (DFA). The following sections describe the specific traces used for the alignment. These are DES and AES traces, and they are known to leak data. The original traces available vary from 500 to 1000 sequences containing 500000 to 980000 samples represented as 1 byte each. The sampling rate for their acquisition was 1GHz. Regarding countermeasures, there are sets with and without them in place. That is also the case for preprocessing steps, such as resampling or which cryptography round operation is the target.

6.2 Analysis Metrics

The results discussed in this section use two main metrics: Pairwise Pearson correlation and Known-key analysis. For the second option, refer back to the background (chapter 2).

Note that Pearson correlation point to us what is roughly aligned. A correlation coefficient of 1 or .9 shows that the power traces are aligned. Nevertheless, in some cases, power traces with a 0.6 correlation coefficient can be used to extract data using known-key analysis.

The important key point in this chapter is that with the proper alignment of power traces, we expect fewer traces to achieve a rank of 1. Note that for known-key analysis (explained in the chapter 2): a rank of 1 means that the recovered value and the known key value for that subkey match. This chapter will focus on only the plots of the Rank evolution from the entire Known-key analysis, as the fragments of where the leakage is happening do not add meaningful information to this analysis.

6.3 Experiments

This section discusses the multiple results that MSA and their differences in the outcome achieves in this work. It starts by discussing MSA results with different parameters and how they impact the alignments. The last part of this discussion is to compare MSA with existing approaches used in the field of embedded security.

6.3.1 MSA Parameters

As discussed before, the MSA method has options such as gap penalties, scoring matrices, and choosing the alignment of DNA/RNA or Protein. These structures have a certain amount of nucleotide representations available. This section shares experiments performed with these parameters and how they impact the final alignment result.

Alphabet Length

The alphabet length determines how many symbols represent the samples of a trace set. If the method uses DNA alignment, these ranges were fixed to 7 (4 symbols plus three ambiguous) as shared in the background chapter2. In the case of proteins, the implementation allows choosing up to 20 symbols representing the amino acids.

The trace in fig. 6.1 is the target by the tests described here. A variation of the symbols representation ranges into amino acids was done and is shown below. The

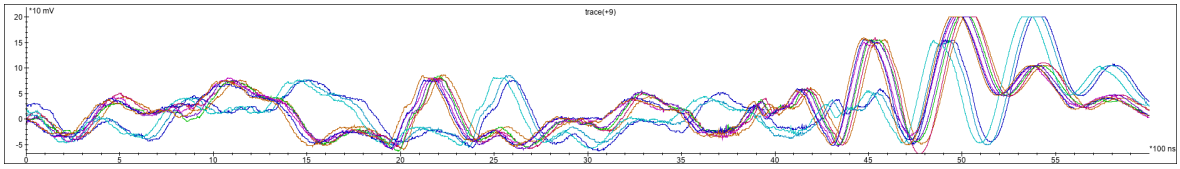


Figure 6.1: Parameters experiments: Source traces

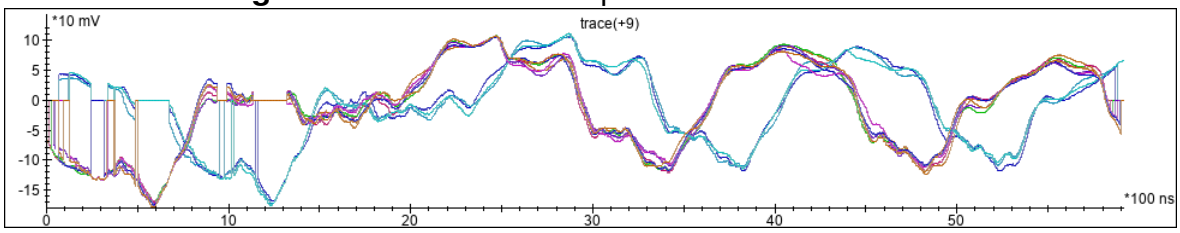


Figure 6.2: MSA aligned: alphabet size 2

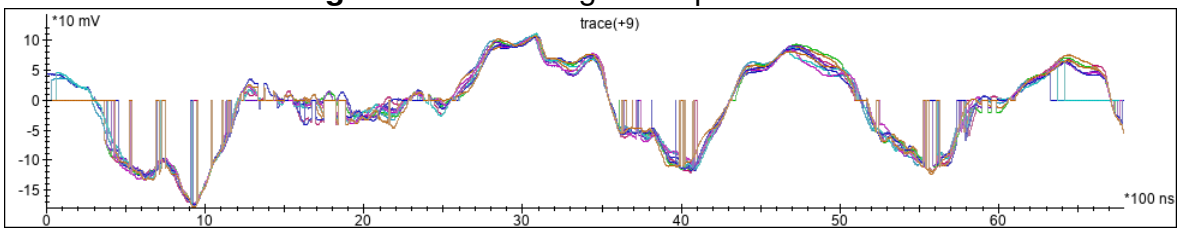


Figure 6.3: MSA aligned: alphabet size 5

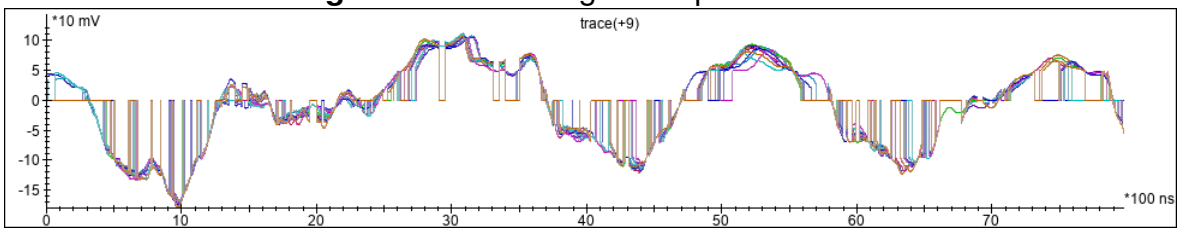


Figure 6.4: MSA aligned: alphabet size 10

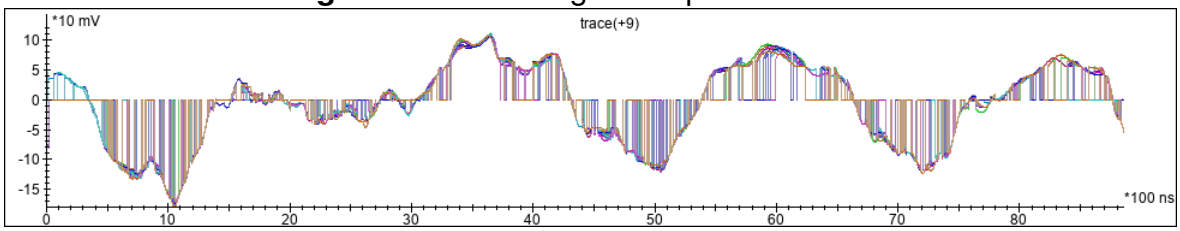
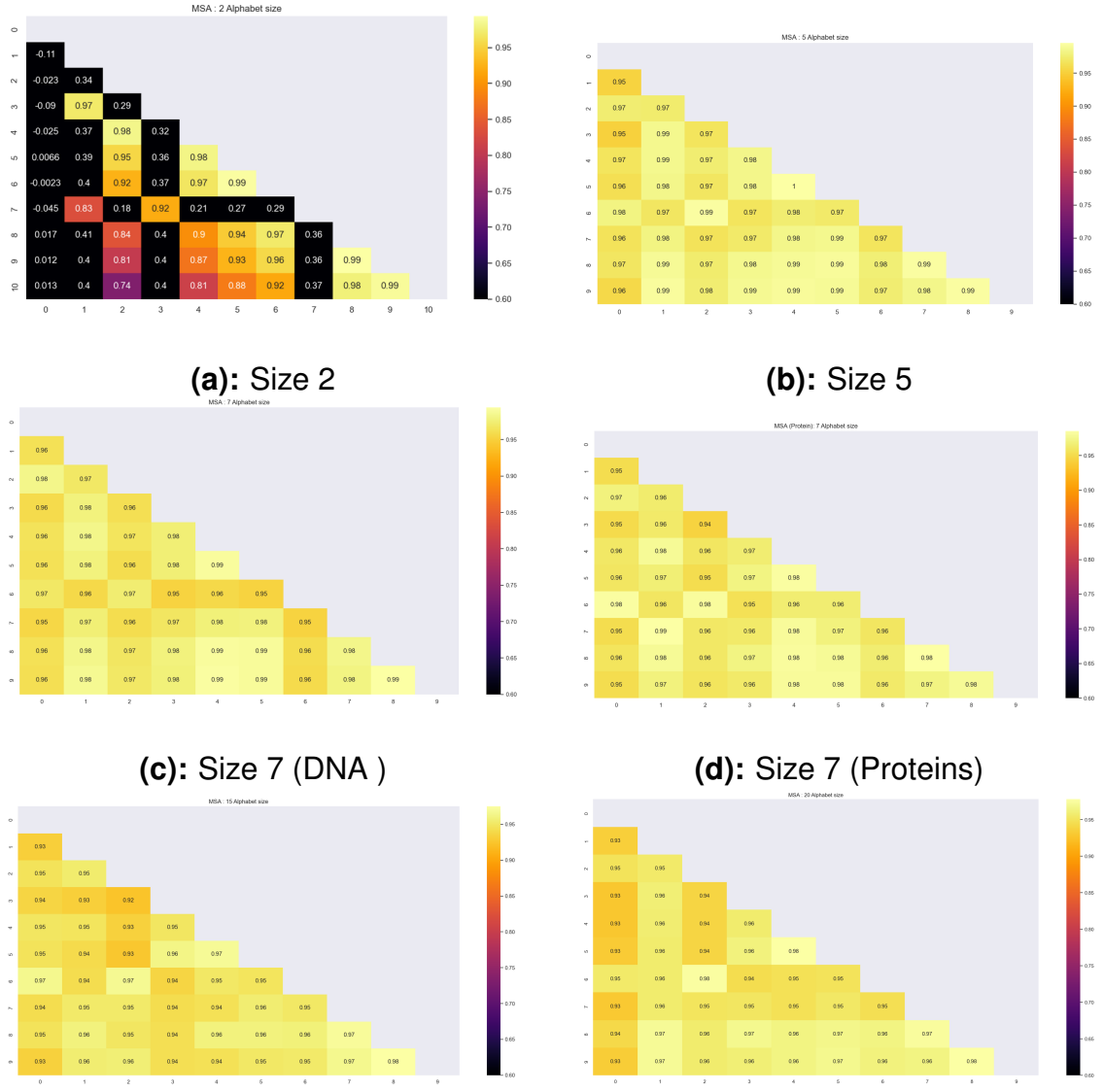


Figure 6.5: MSA aligned: alphabet size 20

proteins examples have the respective order: 2, 5, 7 (DNA and protein), 10 and 20 nucleotide (fig. 6.7, 6.3, 6.4, 6.6c, 6.6d and 6.5). Using two nucleotides is the most comprehensive range possible and does not bring meaningful results, as seen in 6.7. However, the same extreme to 20 nucleotides (fig. 6.5, the smaller samples range amplitude, shows the addition of more gaps. Performance with 20 symbols is also affected, becoming slower. In alignment, more meaningful results start from size 5 to 15 of alphabet length, protein alignment, or DNA. That has to be chosen

by observation of the security analyst as it can vary based on the data collected. Fig. 6.6 shows the pairwise correlation of the post-alignment of 10 traces after using different alphabet lengths: 2, 5, 7, 15, and 20.



(a): Size 2

(b): Size 5

(c): Size 7 (DNA)

(d): Size 7 (Proteins)

(e): Size 15

(f): Size 20

Figure 6.6: Pairwise Pearson-correlation: 10 first traces of the set

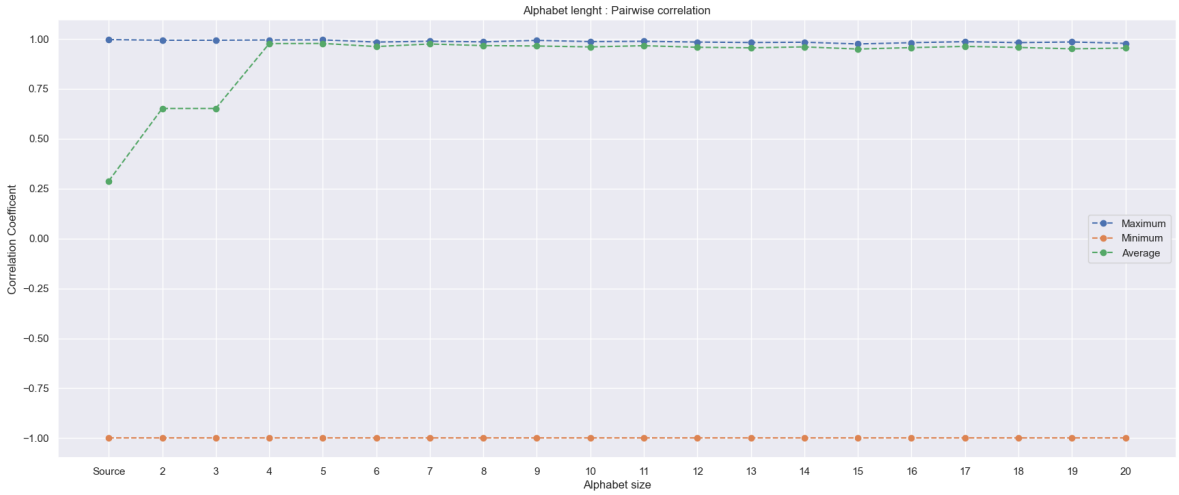


Figure 6.7: Pairwise correlation for all protein alphabet lengths

Gaps conversion method

Alignments done with MSA add ”gaps”. This work has proposed different approaches to interpret these back to samples. Interpreting the added gaps guarantees that the data position shift takes place in the resulting trace and that they align with their reference MSA sequence. This section will investigate the impact on results from representing the gaps using different approaches. The analysis compares two available options: Translating gaps to zero or interpolating an average of the values between gap-opened zones. Figures 6.8 and 6.9 show overlaps of 10 traces with both conversion methods, and the figures 6.10 and 6.11 show an overlap for the eight output bytes of these two traces sets, respectively.

These examples below show the CPA results for the same trace set using the two approaches. Both approaches can deliver meaningful results. The traces used to compare results in the section 6.3.2 are the same as this section uses. This section shares the evaluations of the methods: Gaps transformed into zeros and gaps transformed into an average.

In the fig.6.13 and 6.12, the known-key analysis presents the results for both approaches. For more extensive trace sets, averaging has not performed as well as transforming the gaps to zeros. In some cases generating some higher CPA peaks where they should not exist as the addition of values can create false data-dependent operations and induce false peaks.

Converting the gaps to zeros has proven to be better than averaging these values to eliminate added zeros. Note that fig.6.13 shows the correct value for all sub-keys achieved by 120 traces. That is different when using the average approach. This last approach only finds all keys by 215 traces, approximately.

Global vs. Local Alignment

The option of local [18] and global [17] alignments was also analyzed. The trace set used was the same as in the previous section, containing countermeasures. The leakage appears in fig.6.15 and fig.6.14.

The tests were done on the same machine and with the same parameters for MAFFT using zero-gap penalties, and the traces were represented as DNA sequences with seven symbols. The global alignment performed slightly better than the local alignment option, but in reality, both can still find the sub-keys (See fig. 6.14 and 6.15).

Gap opening penalties

The final alignment traces length depends directly on this parameter, as the MSA adds gaps to its result. This section shows a comparison between alignments that used different gap-opening penalties. The goal is to understand the relationship between the number of gaps inserted and finding the cryptography key. Note that increasing the number of samples in a trace increases the amount of used hard drive space and the processing time of eventual digital signal processing, CPA for example.

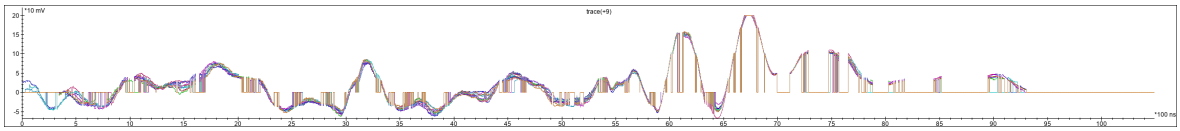


Figure 6.8: Trace using gaps conversion to zero

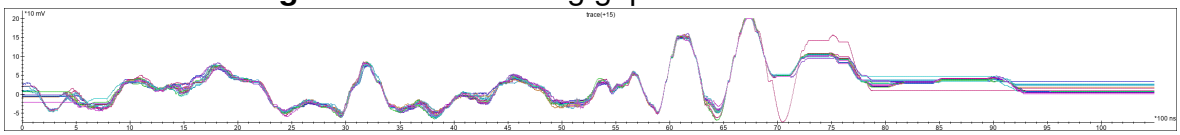


Figure 6.9: Trace using gaps conversion with interpolation

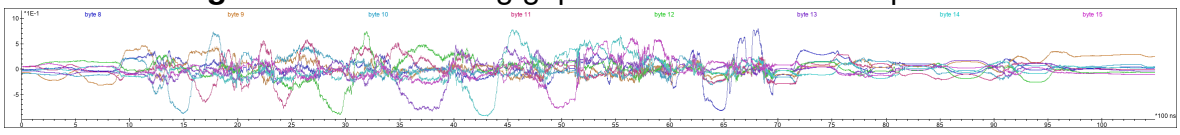


Figure 6.10: CPA: Trace using gaps conversion with interpolation

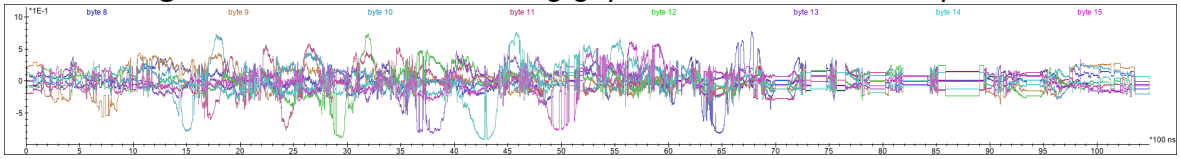


Figure 6.11: CPA: Trace using gaps conversion to zero

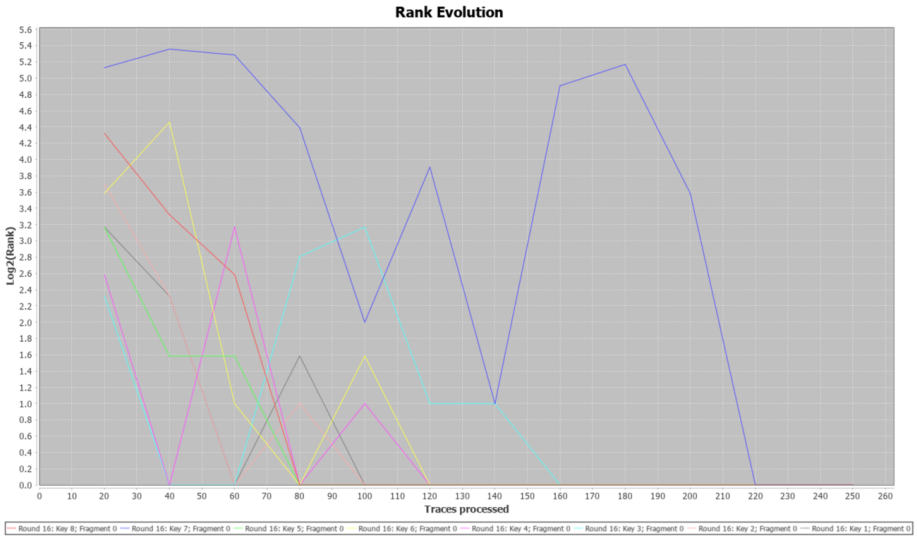


Figure 6.12: Known-key analysis: Trace using gaps conversion averaging samples

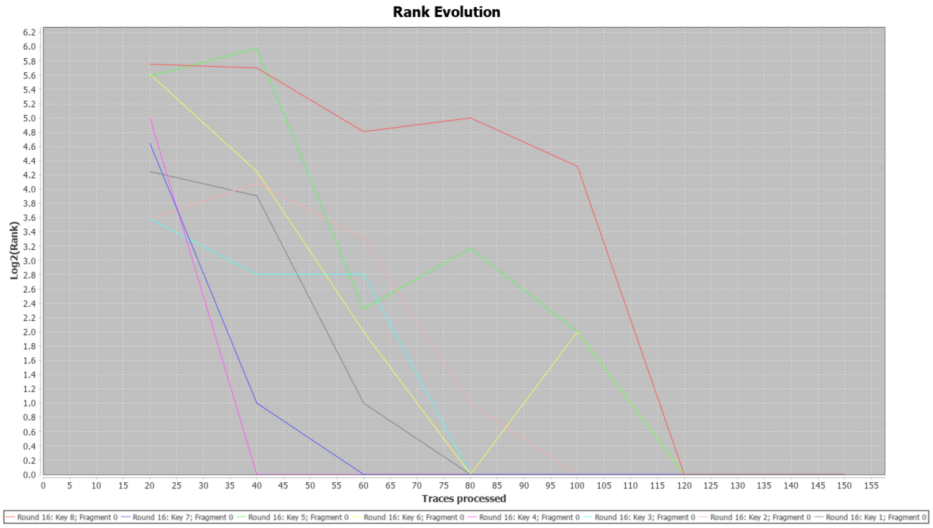


Figure 6.13: Known-key analysis: Trace using gaps conversion to zero

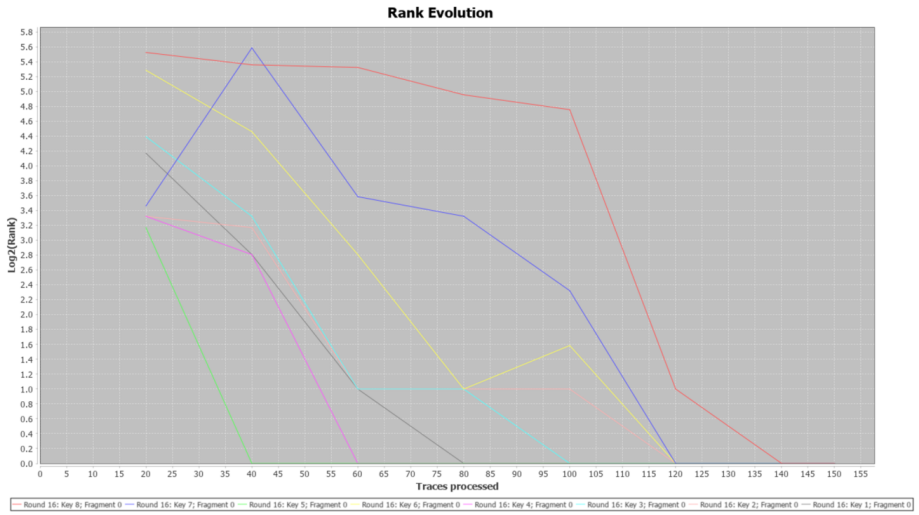


Figure 6.14: Known-key analysis: Local alignment

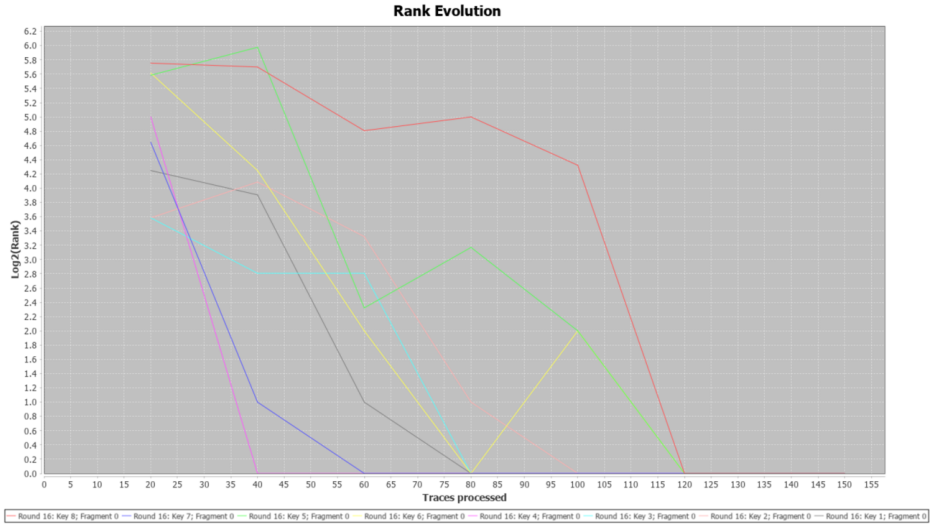


Figure 6.15: Known-key analysis: Global alignment

The original traces in the trace set used for this result had 6000 samples. All alignments performed with MSA have increased the number of samples in all traces, as seen in the table of fig.6.16. As expected, there are fewer samples when the gap penalty is higher. Note that gaps are transformed into zeros or interpolates, adding samples to the final trace in relation to its reference.

Fig. 6.16 shows results. All alignments result from the same 150 source traces. The conclusion was that increasing the gap penalty was not beneficial to the final goal: finding the sub-keys. The results with a higher gap penalty were not as effective in finding the sub-keys. It happens because a higher gap penalty causes more mismatches than a lower one. Mismatches are important in the process. Proper use of this MSA aligns symbols representing closely related regions.

Gap Opening Penalty	0	8	15	26
Trace size	19740	17932	17038	16163
Sub-Keys found (150 Traces)	8	8	6	5

Figure 6.16: Gap opening penalty impact

6.3.2 Comparison of MSA with other methods

This section compares this novel approach with the existing methods (Static and Elastic). There are two main situations investigated, and these are alignments of traces with or without countermeasures. These investigations use different trace sets for each situation. One of the trace sets does not contain countermeasures, and the second part of the section uses a trace set that contains random delays as countermeasures. The traces aligned by MSA in this section have all used zero as gap penalty, global alignment, and DNA as method (Alphabet length 7).

Static [4] and Elastic [9] alignments methods are options to analyze the benefits of the MSA method. Static is the most straightforward and fastest option for alignment methods that do not contain countermeasures, and elastic alignment has shown the best results with static alignment can not be used by strategically stretching and compressing unalignable parts of traces in order to align them.

Traces without countermeasures

The trace set used in this section needs to be aligned due to an imprecise triggering start time, meaning that they differ in time in relation to each other. The main goal of testing with this trace set was to compare our method with the static [4] when the static alignment is the best option. The trace set for the tests here used is a DES operation and contains 1000 traces with 400000 samples(fig. 6.17). Both methods test an attack to the last round of DES(fig. 6.18), and the different approaches' results will be analyzed.

The three options previously discussed were implemented to analyze the best resulting one. The first step aimed to verify if the proposed method could provide an alignment where CPA could detect data leakage and how it would compare to the static alignment [4]. The results are in figures 6.19, 6.20, and 6.21.

From the last round, 100 traces are sufficient (for this specific case) to verify data leakage using CPA. Each trace contains 6000 samples. Three resulting alignments of the original trace set (Fig. 6.18) are in Figures 6.19, 6.20 and 6.21. These traces visual comparison, and their CPA is also shown in the same order in Figures 6.23, 6.24 and 6.25, these plots show the results for the last 8 bytes that represent the output. In all three CPA results, it is possible to see where the output bytes correlate

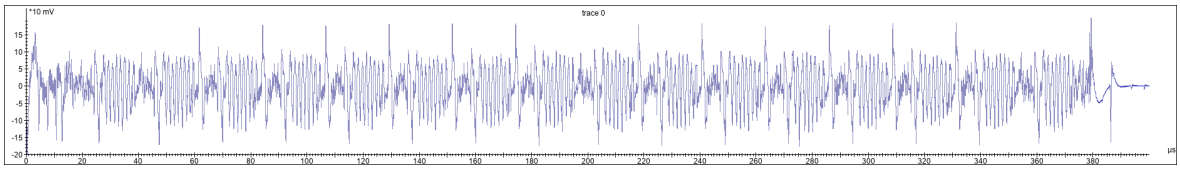


Figure 6.17: DES trace: no countermeasures

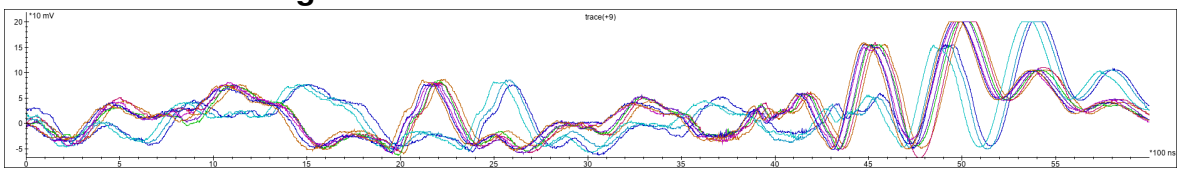


Figure 6.18: DES trace: no countermeasures (Round 16)

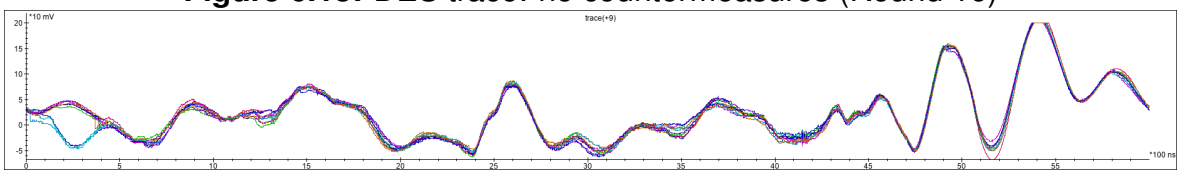


Figure 6.19: Static alignment

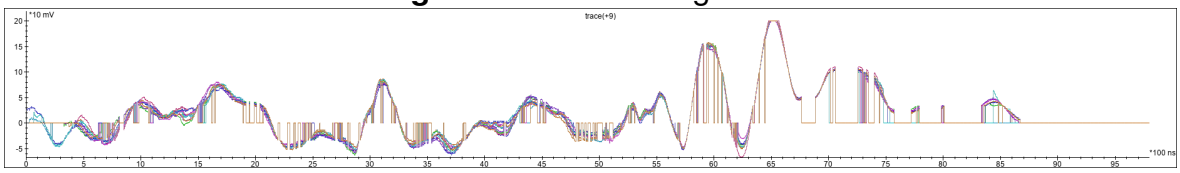


Figure 6.20: MSA alignment with gaps

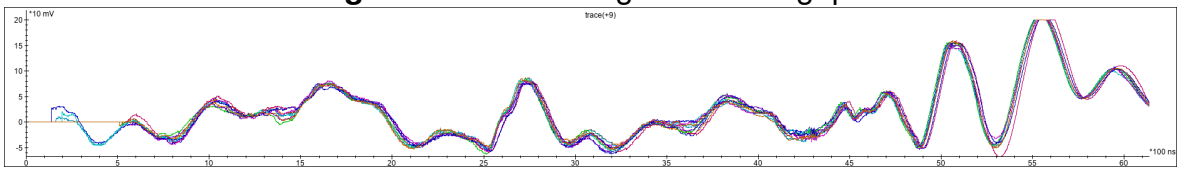


Figure 6.21: MSA alignment gaps to Shift

to the power trace. For comparison with the original trace set, fig. 6.22 shows the resulting CPA of the not aligned region of the traces. It is possible to conclude that all three methods are effective in their result. It is possible to confirm that by comparing the non-aligned traces CPA(fig. 6.22) with the CPA of the aligned traces(fig. 6.23, 6.24 and 6.25). Note that the correlation for the eight output bytes overlapped for all CPA plots.

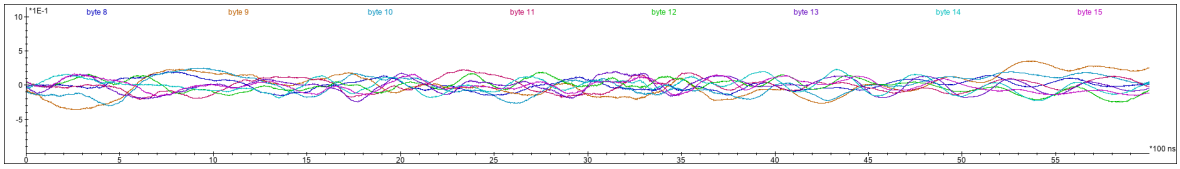


Figure 6.22: CPA: Not aligned original traces

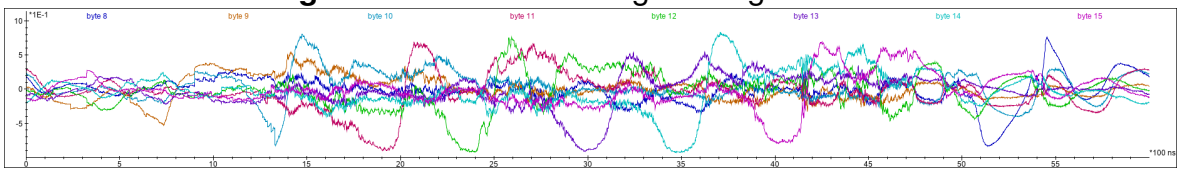


Figure 6.23: CPA: Static alignment

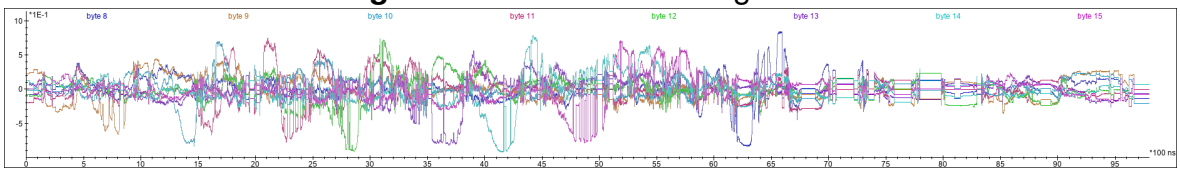


Figure 6.24: CPA:MSA alignment with Gaps

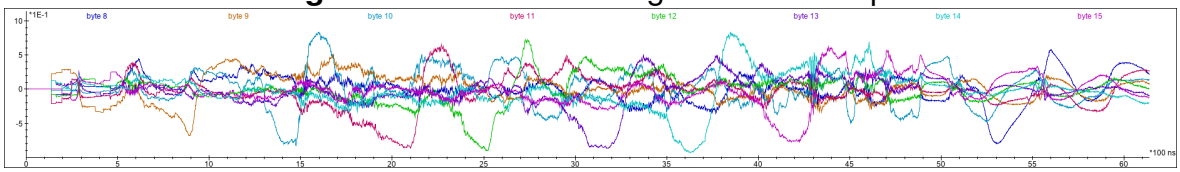


Figure 6.25: CPA: MSA alignment (Static alignment MSA based)

We investigate if the CPA is affected by the fact that extra zeros appear in the power trace when using the MSA approach. It affects the resulting CPA visually in its plot. However, first-order analysis shows no negative impact on finding the sub-keys. The results of using the count of gaps as the shift in Fig. 6.21 had excellent CPA results and were very similar to the static alignment. It is, in essence, the same as static alignment: find the region with the highest correlation in the traces and align them by shifting their difference in time. The alignment can be confirmed with a pairwise correlation between traces of trace set and compared with static alignment. It is possible to achieve an acceptable level of correlation between traces that make CPA successful. The original traces and the two alignments done with the algorithm of MAFFT for MSA and static have its comparison in Fig. 6.26 and 6.27,

these two figures represent a pairwise correlation result of 10 traces in the trace set.

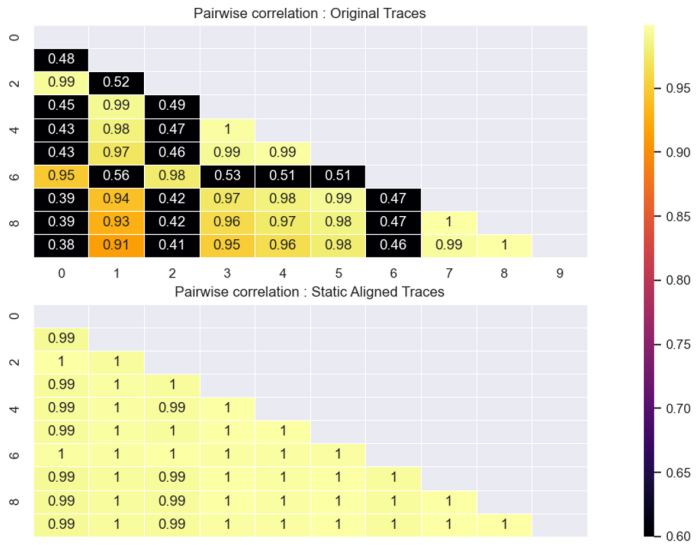


Figure 6.26: 10 Traces pairwise Pearson-correlation: Source & Static Aligned

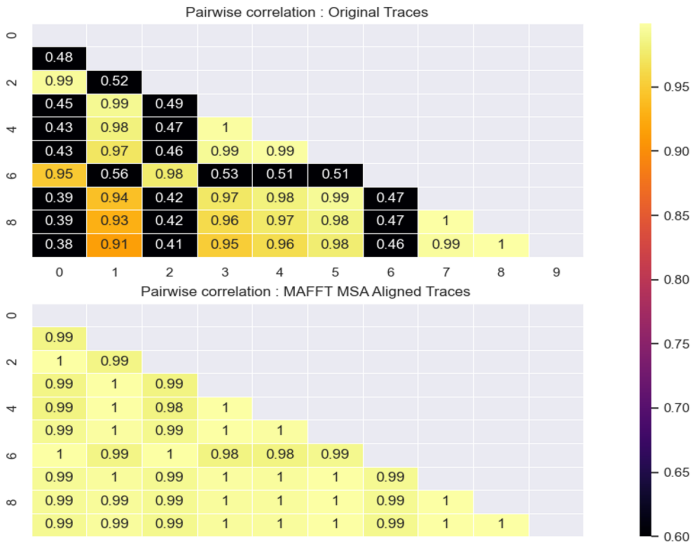


Figure 6.27: 10 Traces pairwise Pearson-correlation: Source & MAFFT MSA Aligned

The tests described here are the first step to better understanding the challenges and results achieved by this work. They also prove the achievement of our goal. The pairwise alignments show considerable improvement in between traces correlation, and the CPA reveals data leakage. This result is very similar to the existing method, with the same amount of traces and samples. Fig 6.28 shows a plot of the CPA result for all methods discussed here using the trace set shown in fig. 6.17.

The MSA method works when only static alignment is needed. Although for this specific case, the improvement of the results does not justify its use.

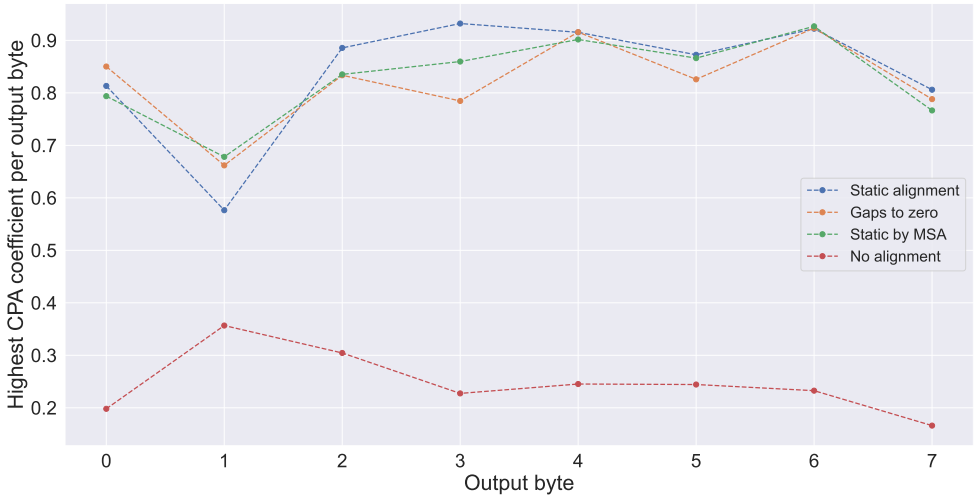


Figure 6.28: DES CPA coefficient comparisons

Traces with countermeasures

The question: ”How can multiple sequence alignment be used for side channel analysis alignments to achieve side channel attacks?” as it is now known, the alignment is possible. This work also explains how to perform it. The sub-questions investigate further the main question bringing more meaningful contributions to the investigation for this novel method :

- If countermeasures are in place, is it possible to minimize their effect on side-channel analysis/attacks using multiple sequence alignment?
- How this alignment compares to existing alignments such as static and elastic?

The traces discussed here are DES operation power traces containing random delays as countermeasures, static alignment [4] is not an alignment solution for them, and success can be achieved with elastic alignment [9]. It is the situation that this work aims to deliver a contribution: Align traces that have countermeasures and achieve performing an attack.

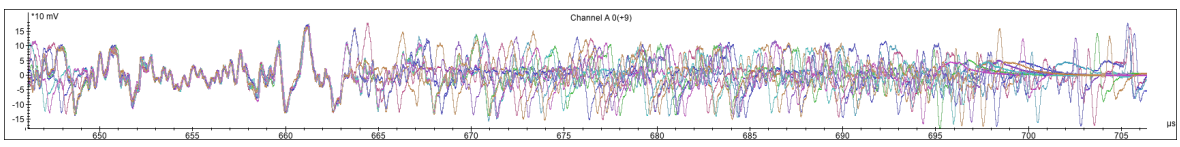


Figure 6.29: Original traces containing random delays (DES round 16)

Fig. 6.29 shows the original set. It has 500 traces with 400.000 samples each. This evaluation approach focuses SCA on the last round of these traces and the

150 first traces of the original trace set. This alignment approach targets between 650 and 663 µs. Observing the picture can help us to understand the challenges of using alignment approaches as static alignment. Aligning at one specific place for this trace does not lead to a successful alignment or a solution to perform an attack. This is an interesting scenario to compare the proposed method with static and elastic alignments.

The trace set used is DES operation focused on the last round of encryption of the previously mentioned 150 traces. The same trace set was aligned with the three methods and analyzed by known-key analysis. These traces contain 6400 samples per trace.

Static Alignment

The first step was investigating how effective or not the static alignment for this trace set is, compared with the existing solution and the proposed method.

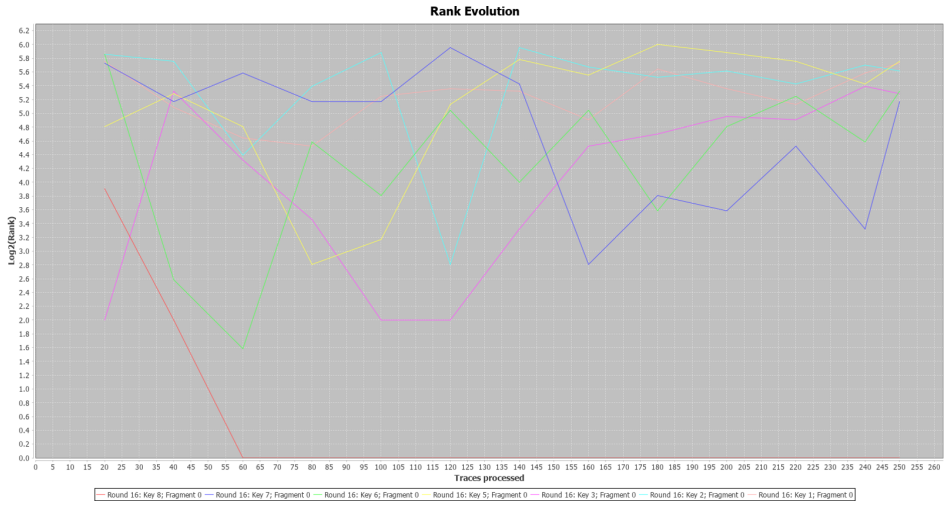


Figure 6.30: Static alignment for random delays

MSA and Elastic Alignment

In this section, both approaches from the previous section test their alignments on the trace set, the elastic alignment, and this works approach. As the MSA includes gaps in the source trace to achieve alignment, the sample number for each fragment is chosen, aiming that both results have the same fragment number. Fig. 6.31 and 6.32 show the results.

The proposed method converges faster, finding at around 140 traces most of the sub-keys (fig.6.31). It has found about half of the sub-keys (fig.6.32).

A comparison of these results is below:

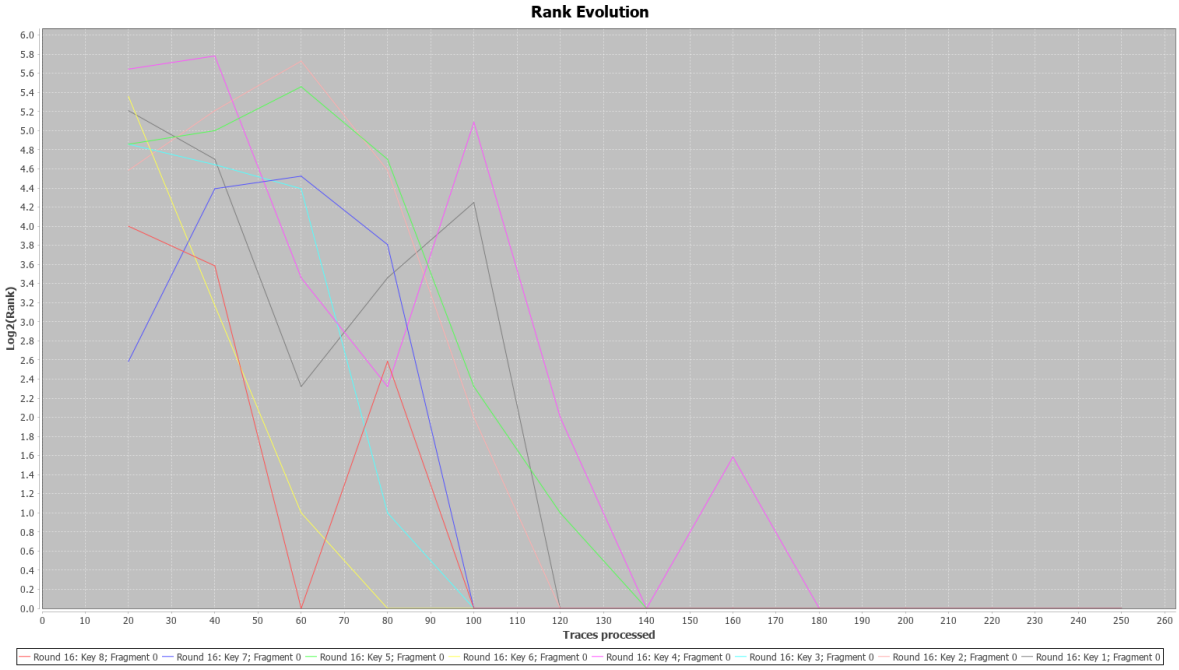


Figure 6.31: Known-key analysis: MSA with random delays

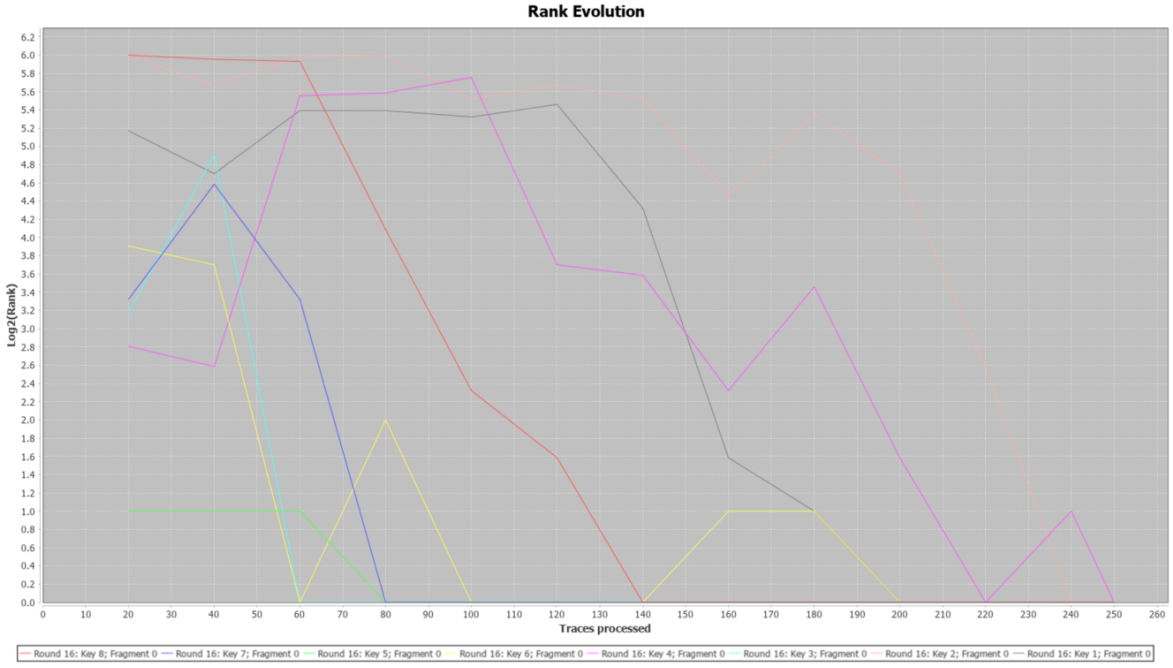


Figure 6.32: Known-key analysis: Elastic alignment with random delays

MSA and AES with countermeasures

In the subsection, we investigate a AES trace set containing random delays. For AES traces sets, CPA coefficient results improve compared to static and elastic alignment; data leakage became more evident with the alignment. Observe fig. 6.33. It proves the versatility of the alignment algorithm that can be a solution to multiple problems. The AES power trace set has 16 bytes of outputs, the fig. 6.33 shows a CPA for these 16 bytes on the last round(10th). MSA has maximized the data leakage achieving higher CPA coefficient for almost all output bytes other than byte 9, which is still very close to what elastic alignment has achieved.

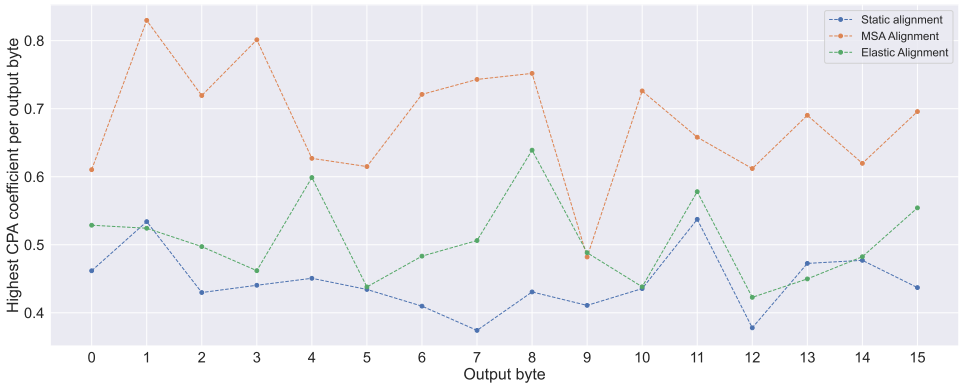


Figure 6.33: AES CPA coefficients

6.3.3 Discussion

The results of the method proposed by this work and the comparison with static and elastic alignment methods are in fig. 6.34 and 6.35.

These tests targeted DES traces that contain random delays as countermeasures. These types of countermeasures can be very effective against attackers using this methodology. Static alignment results show that it does not help much against countermeasures, and it can just find one sub key by 80 traces.

Elastic alignment and MSA found all sub-keys. However, the second method is more effective in finding all subkeys using fewer traces than Elastic alignment. MSA found all the sub-keys with 140 traces, and elastic alignment used 250 traces to achieve the same result.

Sub-Keys Found \ Number of Traces	20	40	60	80	100	120	140	160	180	200	220	240	250
Static	0	0	1	1	1	1	1	1	1	1	1	1	1
Elastic	0	0	1	3	3	3	4	4	4	6	6	7	8
MSA	0	0	0	1	4	6	8	8	8	8	8	8	8

Figure 6.34: Results comparison table

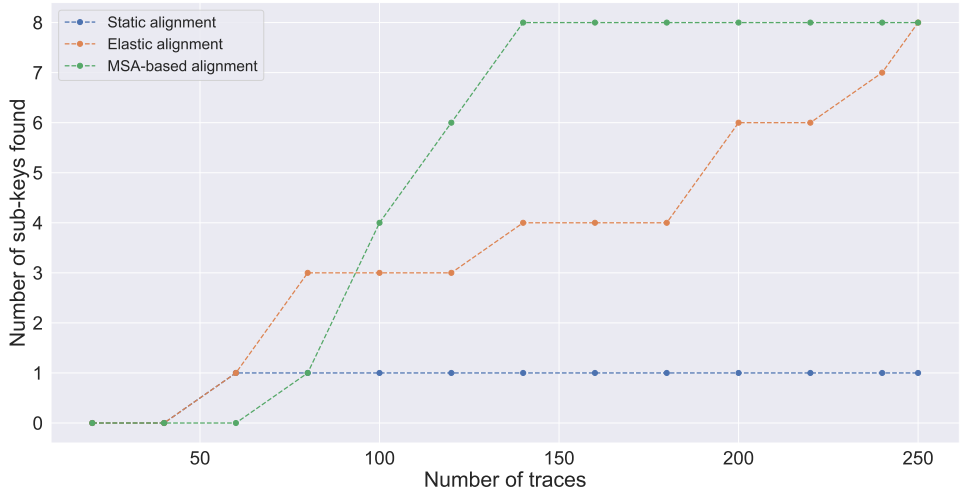


Figure 6.35: Sub-keys found evolution per traces used

6.4 Identifying Random delays

This work also has open possibilities for processing the traces to find meaningful information and identify the random delays. In this section, we discuss what we found.

There are multiple ways to operate with this goal; the benefit of an all to all alignment method such as the one proposed by this work is that regions present in only one trace are isolated. For this example, see the region, 60 to 82 (*100) ns in fig. 6.36 only one of the traces has that region with information that was not gaps (zeros), we can confirm on the consensus curve (fig.6.37) that shows a drop in consensus in the same region.

The implementation as is require input of a threshold: the number of valid samples per column. Assume the user sets 60% as the threshold and the trace set has ten traces. If more than five traces have samples, that column has valid samples. The inverse case is considered the countermeasure. Fig. 6.38 represents a 60% threshold chosen for the ten traces. The figure shows ten traces overlapping each other, and the blue line represents the possible meaningful information. Note that if the blue line is at its minimum, the region is below the threshold. If the above

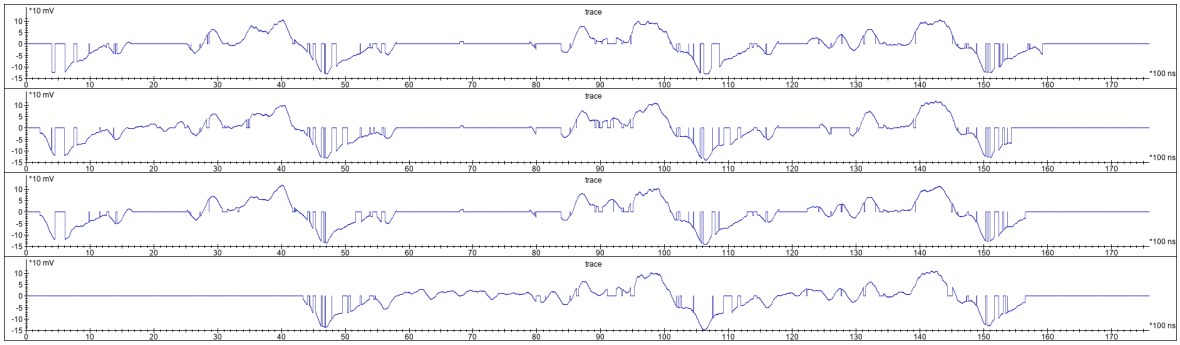


Figure 6.36: DES aligned power traces (MSA)

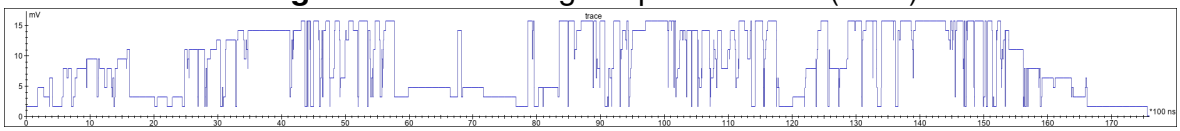


Figure 6.37: DES consensus curve

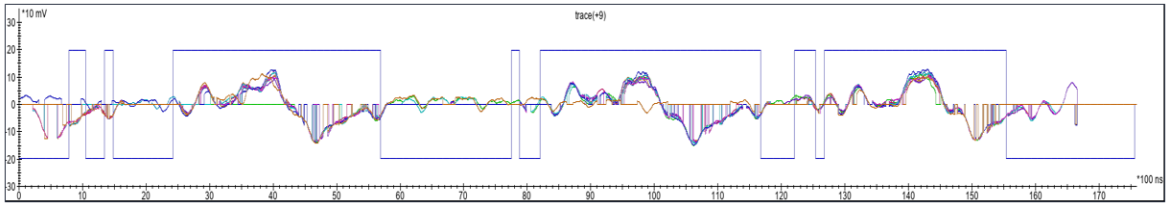


Figure 6.38: DES random delays identification

threshold happens, that represents possible meaningful information. It is up to the user to set suitable thresholds.

We can identify the random delays regions by setting a threshold using this method with AES has similar results. Observe the regions 450 to 510(*100) ns

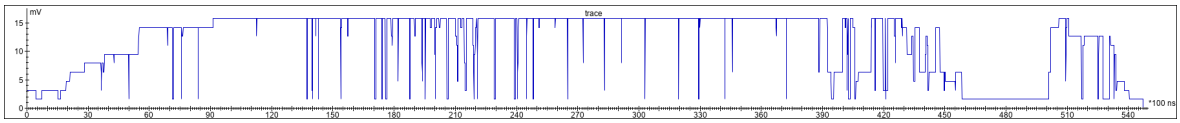


Figure 6.39: AES consensus curve

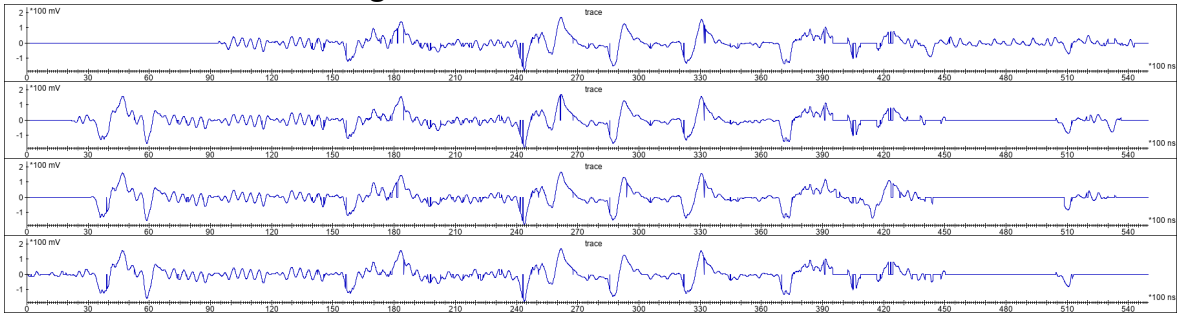


Figure 6.40: AES alignment with random delays

in both fig.6.39 & 6.40, the consensus curve shows the lowest value for that region, and the region has a random delay. However, the region from 30 to 90 (*100) ns has valuable data in most traces and a higher consensus. It happens because the trace patterns were not present on the source traces selected before the alignment.

6.5 Runtime performance

MSA algorithm performance complexity is known to be $O((NM)^{\#Sequences})$, where N and M are the lengths for each sequence. Most of this work was done by transforming singles samples into nucleotide representation. Trace sets used have from 1000 traces with 6000-10000 samples. Fig. 6.41 presents runtime execution times. For reference, Elastic alignment takes about 9.5 seconds to align 250 traces, and static alignment takes 0.3 seconds for the same amount of traces.

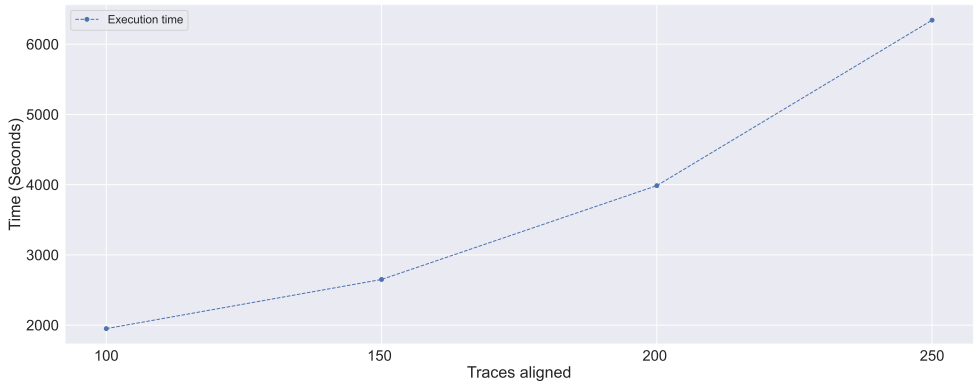


Figure 6.41: MSA execution performance

Chapter 7

Conclusions and recommendations

In this chapter, the conclusion of the project is elaborated over the proposed questions by this research and how this work has achieved to answer to them, main question and sub-questions, therefore concluding this work. Furthermore, the future work section discusses challenges as a way forward for this research and improvement ideas for this method in the Future work subsection.

7.1 Conclusion

This research work presented a novel approach to SCA. This approach comes from an unusual field: genome analysis by bioinformatics. Nevertheless, with the right approach and understanding of how embedded devices' security can benefit from bioinformatics, this work has come up with solutions and ideas that kick-start this joint field research.

This work discussed challenges such as interpreting power traces as DNA or protein sequences and provided implementations. Furthermore, it tested multiple approaches and parameters, exposing their results and revealing the power of this implementation and its strength in future work implementations.

In this section, the questions proposed (main and sub-questions) by the introduction of this work are restated and answered. We are starting with the sub-questions that lead the research to answer the main question more meaningfully. From this point forward, a sub-question will be stated and answered. The first is :

- **Which is the most appropriate performing multiple sequence alignment tool for this research regarding the number of samples and sequences to be aligned?**

This research has investigated several methods for achieving MSA, and these were shared and discussed by the previous sections [21] [22] [47]. The trace sets

used for this research were diverse, varying from 100 to 1000 sequences that could include 6000 to 20000 samples. In these cases, if a conversion is a 1-to-1 conversion, the resulting ”nucleotide” sequence would have the same amount of sequences and characters. Given that condition, MSA methods that could be a solution narrow down to the Clustal family [47] [22] and MAFFT [38]. They both provided similar results when it came to alignments. However, most of this software was available online, and it was possible to upload multiple sequences files and receive the results by e-mail when the alignment was ready. MAFFT was the option that has a Linux software available for download that is part of this work software implementation, which made it possible for automated implementation and performance investigation. For this execution time evaluation purpose, practicality, and lack of significant alignment results difference, MAFFT is the choice.

- **What are the options to convert traces data sets into nucleotide sequences for multiple sequence alignment?**

A step that makes this work possible and meaningful is converting power trace sets (bytes) into characters representing them as a nucleotide. The main benefit to using MSA as an alignment step is that these characters (nucleotides) use a scoring system. Mismatches are possible. With a scoring system, the ”Y” can be considered a ”W”, for example. That was all taken into account in this step.

Several conversion methods were proposed based on this idea discussed here. The first and more effective approach was to define ranges for conversion based on dividing the Y-axis into equal parts. These equal parts are defined by finding the minimum and maximum byte value for the entire trace set and dividing them by the character alphabet length (Maximum 20 for proteins and 7 for DNA). We developed a similar option, and the ranges could be defined as input by the user, meaning that ”Y” could mean a byte value from 0 to 50, while ”W” could be 50 to 65.

The work proposed by [31] as a solution. Multiple other works that need to convert time series into strings similar to this problem used SaX.

Compression options implementations targeted improving performance given the big O complexity of the MSA. Although this has achieved the goal of performance improvement, an already existing re-sampling method was used to compress the power traces. Future work can use compression techniques specific to our methodology.

The approach to compression of the sequences and minimizing the number of sequences was not found and did not fit in the scope of this work. The future work section shares more ideas on this.

- **If countermeasures are in place, how is it possible to minimize their effect on side-channel analysis/attacks using multiple sequence alignment?**

MSA is all-to-all alignment. When mutations in different DNA or protein sequences are in place, it identifies the highest similarity zones and aligns them to each other based on the approaches discussed here. This work took the right approach. It is possible to use that knowledge in this novel method. When looking at the final alignment of the power traces using this approach, they have high similarity zones aligned to one another, and countermeasures do not align to other sample points leading to longer gaps zones in traces with countermeasures.

The consensus function indicates if a region is a mutation or is a meaningful sample, all based on the number of gaps on an x-axis point of our trace set,

- **How this alignment compares to existing alignments such as static and elastic?**

When compared to the static and elastic alignment, this method could achieve the results that both alignment methods could achieve. In situations where only static alignments are required, and there are no countermeasures, both methods achieved similar results. When it came to performance, the method proposed here takes longer time than the static method. However, valuable results appear when the static alignment does not apply, and elastic alignment does, in the presence of random delays as countermeasures. The alignment converged in the confidence rank for finding the known sub-keys with fewer traces than the elastic alignment, showing the power of this approach.

However, this work delivers excellent results regarding the end goal of finding the cryptography keys. This method has succeeded in finding keys with fewer traces than the elastic and static methods, as presented in chapter 6. The elastic method found all the keys with 240 traces, and MSA found the same ones using 120 traces.

Main question:

- **How can multiple sequence alignment be used for side channel analysis alignments to make side-channel attacks more effective?**

This question can be answered by looking at the correlation coefficient improvement when looking at the traces in a pairwise Pearson correlation and by evaluating the ultimate goal of aligning power traces for SCA: revealing secret keys from crypto operations. This question has found positive results in both aligning traces and delivering a final result that is consistent enough to be evaluated with SCA approaches such as CPA. Evaluation of the results has also shown that fewer traces were needed to break sub-keys compared to elastic alignments in the presence of countermeasures. This work has explored multiple approaches, evaluating them to

find the best direction to use MSA for this problem. It is delivering new ideas to the field of embedded systems security.

Although, part of the exploration has expanded, falling out of the scope for the main and sub-questions proposed by this work. The following section discusses more ideas as a way forward for this work.

As final thought and conclusion, this work delivered satisfactory answers for all the questions proposed bringing to life a new possibility of alignments. It also leaves opportunities for further exploring the bioinformatics algorithms for MSA as an option for SCA by leaving comments on possibilities.

Limitations: The MSA algorithm has big O complexity of $O((NM)^{\#Sequences})$, as mentioned by the section 6.5, chapter 6. Trace sets tend to be extensive data converted by this method from a sample to a symbol, one-to-one approach. The bottleneck is then runtime performance. Due to that, alignments were focused on specific rounds of the cryptography operation (round 1 or 16 for DES, for example.). Reducing the samples per trace using a re-sampling preprocessing step can be used as an option. A Field-programmable gate array (FPGA) implementation can be a solution for performance improvement or the methods addressed in the next section for future work.

Furthermore, power traces are large data sets containing thousands of samples. MSA alignments results have the same amount of input samples plus gaps. There is a relationship between trace set size and output sample growth. Aligning 150 traces with 6000 samples each results in 17000 to 20000 samples for each, an increase of over 2.5 to 3 times its size. Storage of these traces demands more disk space, especially in cases where 300 traces of 40000 samples each need alignment. Chapter6 presents these results and their consequences.

Ethics: Providing a new tool to SCA can be potentially harmful when used for illegal reasons. The method proposed provides a new methodology for security operations evaluations. The method is time series alignment, meaning a preprocessing step for actual harmful operations such as DPA and CPA. This work's novel approach has achieved exciting results in extracting secret information that should not be exposed, given the purpose of cryptography.

This work shares its idea to provide an understanding of the approach to a new SCA attack in the hopes that it can protect potential victims from unethical users.

7.2 Future work

Although focused on the answers to the questions initially proposed, this work has also explored multiple options and tested initial approaches that can evolve the ideas presented here in terms of reliability and execution performance. This section opens this discussion and presents what, at the moment, is thought to be exciting solutions as a way forward for the current development.

The biggest challenge in this work is the representation of the power traces into symbols and execution performance. Symbols used with the MSA algorithms represent power traces that are large data sets. Given the alignment algorithm complexity, the time execution has been at times of 25 hours to a larger set of 500 traces containing 10000 samples each. A solution is the re-sampling of the original traces to be able to evaluate results in a shorter period.

This work proved that MSA alignment is a possibility that brings good results and can be valuable research. Performing the alignment in two steps would benefit the final result regarding execution time. Unfortunately, during the time assigned to the development of this research and initial goals, a solution for the first step was not found. The way to improve execution performance is to use the course and granularity approach discussed in Chapter 4.

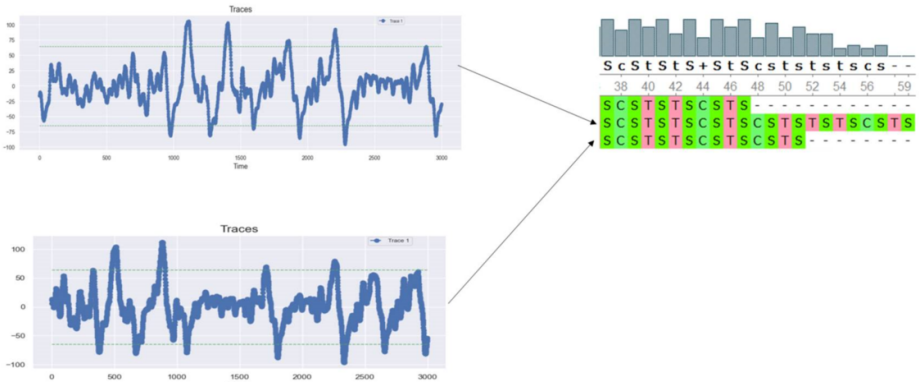


Figure 7.1: Coarse and fine granularity and its conversion to symbols (Step 1)

The first transformation of the traces is done in a coarse approach to achieve the minimum of samples so that the regions can be aligned together and then perform the next step: the granular alignment of similar regions. Fig. 7.1 shows the representation of the traces. It is possible to see that the alignment has multiple possibilities that are not meaningful to power traces alignment, considering just the symbols. This challenge lies in finding an interesting representation for this step's future work. Fig. 7.1 represents the challenge well. Both steps can significantly improve performance as smaller data groups will be aligned.

Apart from execution performance, the length of the alignments results in sequences significantly larger than its input sequences. There is also a challenge with this bottleneck; traces are large data sets, so their alignment will require even more storage memory. As future work, there are also opportunities for improvements. The gaps increase when traces contain too many countermeasures and differ from others in the trace set; the consensus indicates that these sections do not contain meaningful information. Future work can use this information to minimize gaps and power trace zones that do not contain actual data. With the same focus, work that eliminates small open gaps can improve the final trace set size and possibly improve CPA, fig. 7.2 & 7.3 shows both cases. In the first figure, the red circle represents the small gaps that can be closed in future work. The second region shows a red square marking the entire region that does not add to the final result, and elimination of it benefits trace set size reduction.

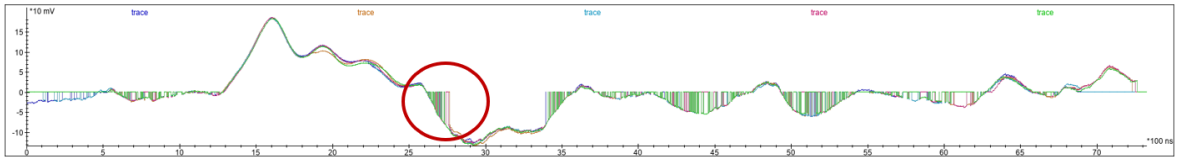


Figure 7.2: Small open gap regions

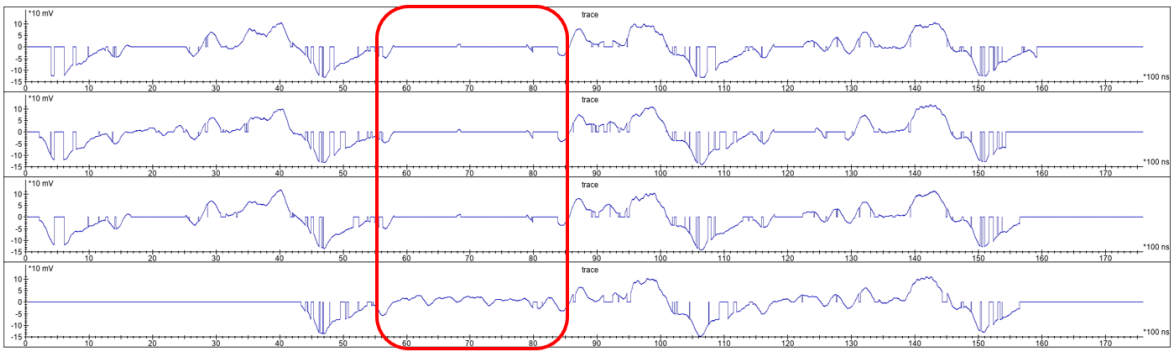


Figure 7.3: Countermeasure gap regions

This work has investigated multiple alignment methods for biological sequences. Remember that they all focus on biological information. Thus, understanding their characteristics benefits this work as part of the conversion steps, such as scoring matrices. For future implementations, it is interesting to implement an alignment algorithm that provides the options of MSA but focuses entirely on time series implementing gap penalty options and its scoring matrices that benefit their alignments, for example.Although focused on SCA, This research can offer more possibilities in fields that need alignment of time series and understanding of the variations and how they relate to each other.

Acknowledgements

This work represents a life step I could never imagine five years ago; life has surprises. It all started by leaving the United States of America for The Netherlands in the middle of the worst COVID pandemic time in terms of restrictions with a mix of excitement and concern. Decision many would never take or understand, but now I know it has just so much learning to all of it that I have no regrets about taking it. This beautiful life step ends by presenting this work written in this report.

I want to thank dr. ir. Nikolaos Alachiotis and Vipul Arora. The guidance throughout the project, the time devoted to assisting me, the openness to all sorts of questions in different domains I had (Oh boy, they were many...), your excitement with every step, the freedom I have been given to create and the knowledge sharing has made this an incredible journey to work with you both. I will never forget all the learning our relationship during this project has brought me.

Thanks to my life partner, Marieke. For giving me the courage to transform a plan into action and being the main trigger of my decision to change, taking this step towards a fulfilling life in many domains. Also, to her family, that took me to their own family in The Netherlands.

Nobody gets to this point without a solid base, and I have my mom, Helena, to thank. Thank you for your lifelong hard work and care to always raise Vinicio and me to your best possible.

To all my lifelong close dear friends who are now all over the planet, we still have close contact almost daily (Arthur, Matheus, Livia, Vando, Yuri, Thiaguete, DPdex, and Cachina) and gave me much emotional support. Although a younger friend, Adelson who, inspired me by always sharing his Ph.D. process with passion and was always up for 25km random runs through the most inconvenient paths in Overijsel. Thanks, guys.

Thank you!

”Two roads diverged in a wood, and I
I took the one less traveled by,
And that has made all the difference.”
*Extract from The Road Not Taken by Robert Frost

Bibliography

[1] M. Matsui, “Linear cryptanalysis method for des cipher,” in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1994, pp. 386–397.

[2] T. Jithendra, K.B.; Shahana, “Enhancing the uncertainty of hardware efficient substitution box based on differential cryptanalysis,” *In Proceedings of the 6th International Conference on Advances in Computing, Control, and Telecommunication Technologies (ACT 2015), Trivandrum, India*, vol. 45-B, p. 318–329, October 2015.

[3] N. T. Courtois, “Feistel schemes and bi-linear cryptanalysis,” in *Annual International Cryptology Conference*. Springer, 2004, pp. 23–40.

[4] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer Science & Business Media, 2008, vol. 31.

[5] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.

[6] D. Brumley and D. Boneh, “Remote timing attacks are practical,” *Computer Networks*, vol. 48, no. 5, pp. 701–716, 2005.

[7] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” in *International workshop on cryptographic hardware and embedded systems*. Springer, 2004, pp. 16–29.

[8] C. Clavier, J.-S. Coron, and N. Dabbous, “Differential power analysis in the presence of hardware countermeasures,” in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2000, pp. 252–263.

[9] J. G. van Woudenberg, M. F. Witteman, and B. Bakker, “Improving differential power analysis by elastic alignment,” in *Cryptographers’ Track at the RSA Conference*. Springer, 2011, pp. 104–119.

[10] M. Chatzou, C. Magis, J.-M. Chang, C. Kemena, G. Bussotti, I. Erb, and C. Notredame, "Multiple sequence alignment modeling: methods and applications," *Briefings in bioinformatics*, vol. 17, no. 6, pp. 1009–1023, 2016.

[11] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, "Report on the development of the advanced encryption standard (aes)," *Journal of research of the National Institute of Standards and Technology*, vol. 106, no. 3, p. 511, 2001.

[12] M. Randolph and W. Diehl, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptography*, vol. 4, no. 2, p. 15, 2020.

[13] A. Shamir, "Protecting smart cards from passive power analysis with detached power supplies," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2000, pp. 71–77.

[14] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards." *Smartcard*, vol. 99, pp. 151–161, 1999.

[15] C. Whitnall and E. Oswald, "A fair evaluation framework for comparing side-channel distinguishers," *Journal of Cryptographic Engineering*, vol. 1, no. 2, pp. 145–160, 2011.

[16] "Riscure inspector." [Online]. Available: https://www.riscure.com/security-tools/inspector-sca/

[17] C. D. Needleman, Saul B. & Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48 (3), p. 443–53, 1970.

[18] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[19] M. Dayhoff, R. Schwartz, and B. Orcutt, "22 a model of evolutionary change in proteins," *Atlas of protein sequence and structure*, vol. 5, pp. 345–352, 1978.

[20] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks." *Proceedings of the National Academy of Sciences*, vol. 89, no. 22, pp. 10 915–10 919, 1992.

[21] "Mafft." [Online]. Available: https://mafft.cbrc.jp/alignment/server/index.html

[22] "Clustalw." [Online]. Available: https://www.genome.jp/tools-bin/clustalw

[23] "T-coffee." [Online]. Available: https://tcoffee.crg.eu/

[24] “Ugene.” [Online]. Available: http://ugene.net/

[25] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[26] R. A. Muijrers, J. G. van Woudenberg, and L. Batina, “Ram: Rapid alignment method,” in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2011, pp. 266–282.

[27] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[28] C. Clavier, J.-S. Coron, and N. Dabbous, “Differential power analysis in the presence of hardware countermeasures,” in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2000, pp. 252–263.

[29] D. Strobel and C. Paar, “An efficient method for eliminating random delays in power traces of embedded software,” *H. Kim (Ed): ICISC 2011, LNCS 7259,Springer-Verlag Berlin Heidelberg 201*, pp. 48–60, 2012.

[30] J. Tarhio and E. Ukkonen, “Approximate boyer–moore string matching,” *SIAM Journal on Computing*, vol. 22, no. 2, pp. 243–260, 1993.

[31] Y. Yu, Y. Zhu, D. Wan, H. Liu, and Q. Zhao, “A novel symbolic aggregate approximation for time series,” in *International Conference on Ubiquitous Information Management and Communication*. Springer, 2019, pp. 805–822.

[32] J. Lin, E. Keogh, L. Wei, and S. Lonardi, “Experiencing sax: a novel symbolic representation of time series,” *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[33] J. Lonardi and P. Patel, “Finding motifs in time series,” in *Proc. of the 2nd Workshop on Temporal Data Mining*, 2002, pp. 53–68.

[34] F. Lemoine, L. Blassel, J. Voznica, and O. Gascuel, “Covid-align: Accurate online alignment of hcov-19 genomes using a profile hmm,” *Bioinformatics*, vol. 37, no. 12, pp. 1761–1762, 2021.

[35] R. A. Muijrers, J. G. van Woudenberg, and L. Batina, “Ram: Rapid alignment method,” in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2011, pp. 266–282.

[36] C. Guo, H. Li, and D. Pan, “An improved piecewise aggregate approximation based on statistical features for time series mining,” in *International conference on knowledge science, engineering and management*. Springer, 2010, pp. 234–244.

[37] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding *et al.*, “Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega,” *Molecular systems biology*, vol. 7, no. 1, p. 539, 2011.

[38] K. Katoh, K. Misawa, K.-i. Kuma, and T. Miyata, “Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform,” *Nucleic acids research*, vol. 30, no. 14, pp. 3059–3066, 2002.

[39] J. D. Thompson, D. G. Higgins, and T. J. Gibson, “Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice,” *Nucleic acids research*, vol. 22, no. 22, pp. 4673–4680, 1994.

[40] C. Notredame, D. G. Higgins, and J. Heringa, “T-coffee: A novel method for fast and accurate multiple sequence alignment,” *Journal of molecular biology*, vol. 302, no. 1, pp. 205–217, 2000.

[41] R. C. Edgar, “Muscle: multiple sequence alignment with high accuracy and high throughput,” *Nucleic acids research*, vol. 32, no. 5, pp. 1792–1797, 2004.

[42] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[43] “Riscure trs library.” [Online]. Available: https://trsfile.readthedocs.io/en/latest/

[44] D. J. Lipman and W. R. Pearson, “Rapid and sensitive protein similarity searches,” *Science*, vol. 227, no. 4693, pp. 1435–1441, 1985.

[45] “Time series alignment with msa.” [Online]. Available: https://github.com/HectUch/python-timeseries_MSA

[46] “Riscure piñata board.” [Online]. Available: https://www.riscure.com/products/pinata-training-target

[47] “Clustalomega.” [Online]. Available: http://www.clustal.org/omega/