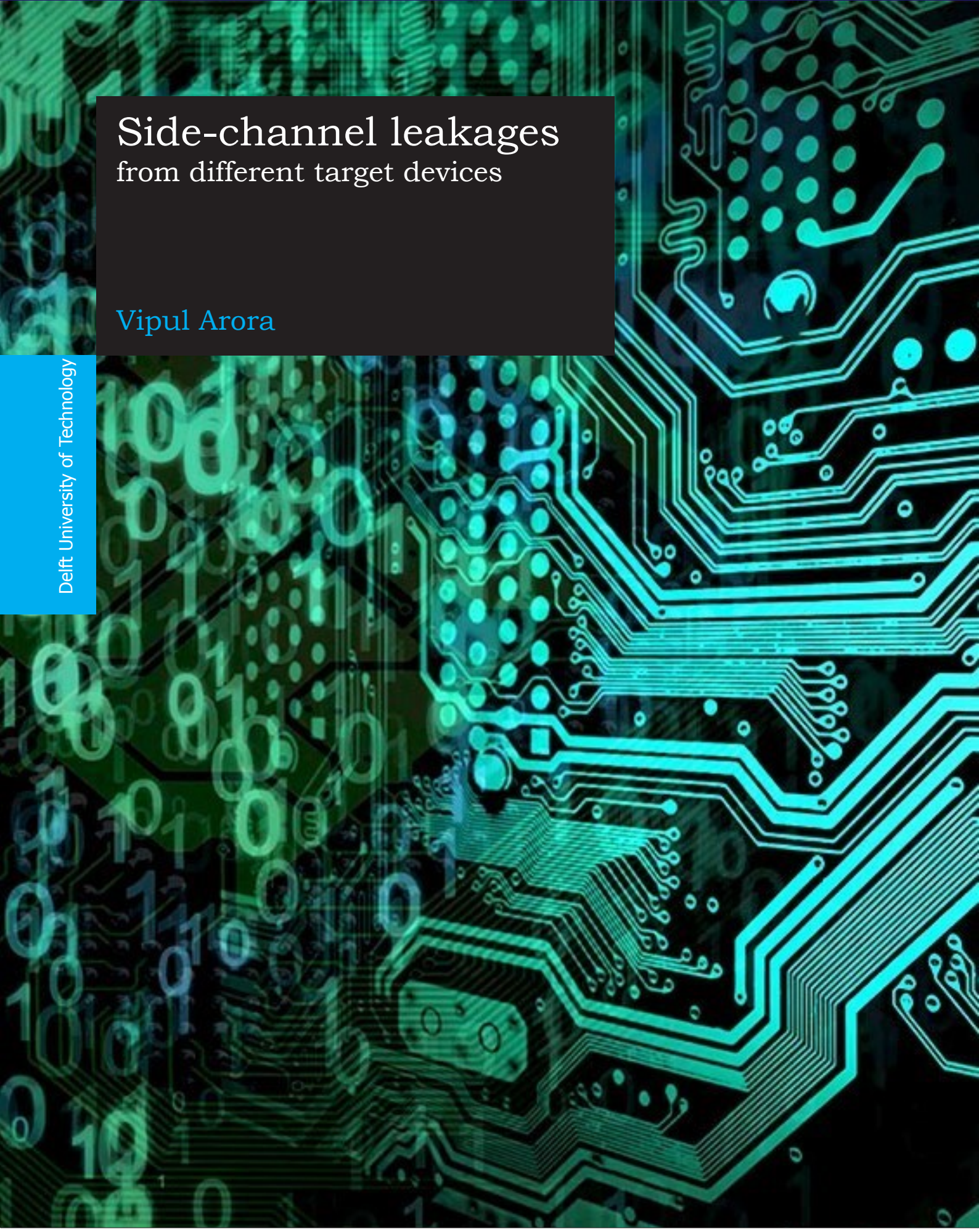


Side-channel leakages
from different target devices

Vipul Arora

Delft University of Technology

TU Delft Delft University of Technology

Challenge the future

Side-channel leakages

from different target devices

by

Vipul Arora

in partial fulfillment of the requirements for the degree of

Master of Science
in Embedded Systems

at the Delft University of Technology,
to be defended publicly on Tuesday December 15, 2020 at 8:45 AM.

Student number:	4770269	
Thesis committee:	Prof. dr. ir. R.Ł. Lagendijk,	TU Delft, Chair
	Dr. ir. S. Picek,	TU Delft, Supervisor
	Dr. ir. I. Buhan,	Radbound University
	Dr. ir. M. Taouil	TU Delft

This thesis is confidential and cannot be made public until December 15, 2020.

TU Delft Delft University of Technology

Abstract

The advances in cryptography have enabled the features of confidentiality, security, and integrity in the digital space. The information about the working of the digital system is used to perform side-channel attacks. These attacks exploit the physics of the system rather than targeting the mathematical complexity of algorithms. Side-channel attacks measure the variations in the system's physical characteristics to obtain information about the operations being performed along with the operand data.

In this work, we evaluate how the choice of physical target device impacts the cryptographic implementation's security. A software implementation is flashed on devices from two different manufactures with the same instruction set, configured for identical execution. Power traces from different hardware devices are acquired and evaluated using leakage detection methodologies of TVLA, and KL-Divergence. Trace-sets are compared at the abstraction level of intra-board, inter-board, and inter-class to explore the information leaks. The performance of leakage detection methodologies in identifying leaks is evaluated using key-rank analysis and verified by profiling templates.

Results show two classes of devices belonging to different manufacturers vary significantly in terms of the power profile yet show similarities in data leakage. Based on the source of leaks; micro-architecture leaks have minor differences at the inter-board level within boards of the same class, though the results of micro-architecture leaks are not comparable across boards of different classes. Data-overwrite leaks are specific to the instruction set and pipeline implementation and are observed for both classes of devices. This work provides a methodology for evaluating software implementations across different hardware.

Preface

Before you lies my thesis, which marks the completion of my masters at the TU Delft. I started the thesis in January 2020 while the world was still normal. However, switching to working from a room and meeting people over video calls did impact productivity. This year-long journey had its highs and lows with a lot of learning. There are many people who directly or indirectly contributed to this work, I would like to take this opportunity to thank them.

Firstly, I would like to thank my supervisor, Stjepan, for guiding me through a big part of my master's. You helped me out with the ever-changing situations and were accommodating for meetings, even at non-typical timings. Thank you for your support, which has enabled me to fulfill this achievement. I would also like to thank Ileana for her guidance in making sense of the story and proofreading. Thank you for spontaneous meetings and for mentoring me throughout the thesis.

I am thankful to Riscure for giving me the internship and allowing me to take the laboratory setup at home. I would like to thank Guilherme and Baris for taking an interest in my project and providing their feedback to direct the project.

Finally, I am thankful to Shruti, Fanis, Greg, and Jure for being able to distract me when I wanted to relax over breaks. I am grateful to my parents for supporting me through my studies, without whom this would not have had been possible.

Vipul Arora
Delft, December 2020

Contents

1

Introduction

In recent years growth in large scale semiconductor manufacturing has enabled the fast development of smart devices. An application developer does not need to bother with processors' hardware design to develop applications for most cases. These applications can be sensitive, though not all product development teams have resources for conducting security evaluations.

The advances in cryptography have enabled the features of confidentiality, security, and integrity in the digital space. The current widely used encryption algorithm AES (1998) has proven to be computationally complex for brute force attacks on the key, which can be assumed secure against classical adversaries with time and memory constraints [1]. Using crypto algorithms, developers design complex systems with confidence, assuming the secrecy of the key. In practice the hardware used for implementing the algorithm also needs to be considered as a part of the attack surface to be protected against adversaries with physical access.

Academic works have proven attacks on real-world devices such as the KeeLoq remote entry system [2], the bit-stream encryption in Xilinx FPGAs [3] and Mifare DESFire contactless payment cards [4]. These works utilize the information from the system's side-channel to expose the secret that can be employed for reverse-engineering the system, IP thefts, or injecting trojans.

The information obtained about the working of the underlying system is used to perform **side-channel attacks**. These attacks exploit the physics of the system rather than targeting the mathematical complexity of algorithms. Side-channel attacks measure the variations in the system's physical characteristics to obtain information about the operations being performed along with the operand data. Relating this information, an adversary can deduce the private key data of a cryptographic implementation and breach the system's security. It is common knowledge in security that a system is as strong as the weakest link. The security of a cryptographic system is likewise dependent on the processor's underlying implementation performing these operations [5]. If information about the sensitive data (i.e. password or key data) can be deduced by observing the physical characteristics of the system, the system is said to have **leaks**[1].

Encryption/Decryption involves performing a sequence of mathematical operations on the input and output data, respectively, utilizing a secret-key. When coded in software and implemented using an **instruction set** of a general-purpose CPU, these mathematical operations are termed as **software implementation**. Some cores have dedicated logic circuitry for performing the cryptographic operation at a semiconductor level and is termed as a **hardware implementation**. Irrespective of the type of implementation, hardware or software,

[1]The term leaks refers to leakage of information

an adversary can obtain knowledge about the key data while sensitive operations requiring manipulation with key data are performed. The observations can be recorded in the form of power or electromagnetic traces, referred to as **traces** in short. In side-channel analysis, these traces are used to make deductions on the sequence of the operations and the operand data used.

The early works in side-channel analysis treat the target device as black-box to extract information pertaining to **cryptographic primitives** [6]. To develop countermeasures the works have focused on adding randomness and masking the leaks [7–9]. It has been a cat and mouse game of developing new countermeasures and attacks for the last two decades. More recent works have been focused on generating complex power models [10] and employing deep learning techniques to predict the key from trace-sets [11]. The research direction detached away from leakage sources at the physical layer and focused more on attack methodologies. Our work takes a different direction and focuses on the effects of hardware choices on leakage.

An adversary collects multiple traces while repeating the target operation on varying input data. The power traces, labeled with the corresponding input and output, are recorded to create **trace-sets**. **Differential power analysis** is an advanced form of side-channel analysis, using statistical tools to compare the deviations between power-traces relating the input and output data for predicting the secret-key data. Differential power analysis has existed for about 20 years now, first presented by Kocher et al. [12].

The research question we ask in this thesis is both simple and practically relevant for an embedded system developer assigned to implement an existing cryptographic algorithm (e.g tiny AES, Mbed TLS) on a micro-controller family (e.g ARM Cortex-M0, a popular choice in the IoT industry). The developer is free to choose a micro-controller meeting the project's specifications from the diverse SoC range offered by different manufacturers (ST, NXP, Nordic, etc.). The devices supporting a similar instruction set (e.g ARMv6), can vary in design; firstly based on the semiconductor technology used in manufacturing, which influences the physical characteristics of a device, and secondly, as a function of the changes in the HDL schematic of the micro-architecture to achieve the functional features desired for the specific device. The evaluation is done on hardware at different abstraction levels as shown in figure 1.1; devices from different manufacturers classified as **inter-class**; board iterations from same manufacturer are classified as **inter-board**; varying key and input data on the same physical board is classified as **intra-board**.

In this work, we evaluate how the choice of the physical target device impacts the security of the implementation of the cryptographic algorithm. Concretely, the questions relevant to our embedded system developers are:

1. How significant are the inter-key and inter-data variations?
2. How significant are inter-device manufacturing variations?
3. What is the impact of micro-architectural differences across manufacturers, on the SCA leakage for a given software implementation?

To answer these questions, we use **Statistical analysis** along with leakage detection methodologies to compare trace-sets at different levels. Statistical analysis is performed until the 3^{rd} order, namely based on **mean**, **standard deviation** and **skew**, which gives us an idea about the probability distribution of the trace-set. **TVLA** methodology, **KL-Divergence** and **key-rank analysis**, are the statistical tools used to identify leaks in the trace-sets. Hamming weight model is used for profiling the trace-set to generate template models, which are used to comment on the source of leaks and portability of attacks. Doing so, this works

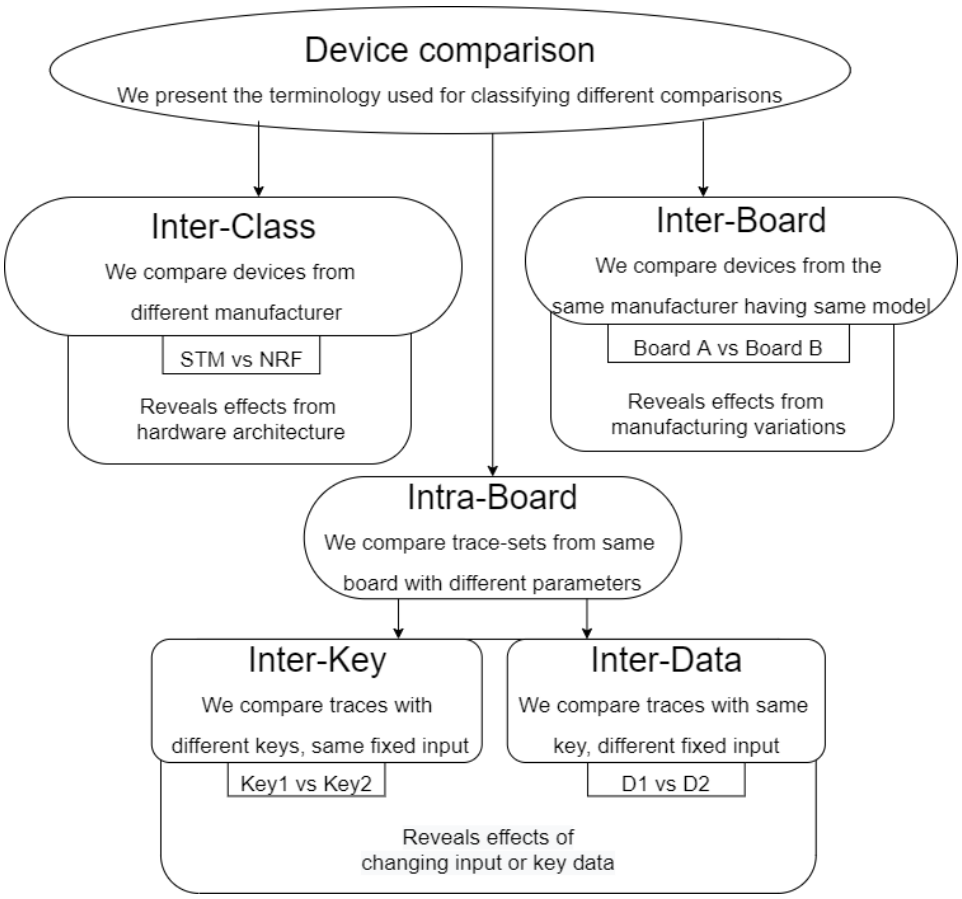


Figure 1.1: Different abstractions of device comparison over which the proposed methodology is used

provides a framework for extensive side-channel evaluation to compare software crypto implementations, highlighting the impact of micro-architecture on data leaks.

Devices belonging to the same family[2] have differences in data leaks. For a class of devices[3], two board iterations differ from manufacturing process variations. The instruction traces[4] from devices of a class are of the same shape, though we observe each physical board to have its signature. The signature here is composed of the deviations in amplitude, specific to sections of instruction trace (ie-Board A has higher amplitude for the first half of the clock while executing ALU instruction compared to board B). Comparing two physical boards, we have consistency in how the difference of means varies along the instruction trace. The same behavior is observed for information leaks; the leakage results can be clustered based on the physical board from which trace-set is acquired(ie-board A shows higher amplitudes of leakage for a specific section of `STR` instruction); a board may be leakier for specific sections of instructions. The influence of key and input data on the results of leakage detection methodologies is presented to highlight the distinguishing features of these measures. Lastly, a methodology for selecting points of interest to profile templates for portability across devices with the same software crypto implementation is presented. This is the first known case of template attack across boards from different manufacturers, to the best of our knowledge, on an unsecured AES implementation.

The work done in this thesis has a wider application than the practical relevance for the

[2]Devices with the same instruction set architecture, manufactured by different vendors
[3]Devices from the same manufacturer with the same model number
[4]trace acquired for the duration over which the instruction is being executed

above-mentioned embedded system engineer. First to the area of leakage simulators, which face the problem of portability across devices even with the same hardware architecture. ELMO [13] authors provide a leakage model for STM devices, though using ELMO on other Cortex M0 devices require profiling instruction triplets from the target device before use.

Secondly to the area of deep learning for SCA, as shown by X-DeepSCA [14] on the portability of template attacks across different physical boards. Unlike other works where profiling and attack traces are from the same target board, X-DeepSCA successfully create a DNN trained over multiple boards' traces to have a Cross-board SCA attack. The work done in this project takes one step further and can be used as input to the problem of creating an NN capable of attacking implementations from the same family of devices.

The remainder of this report is structured as follows: Chapter 2 provides an overview of hardware architecture, assembly AES implementation, side-channel leaks with discussion on leakage detection methodologies. In Chapter 3, the state of art research relating the leaks from devices to their hardware architecture is presented. The work done as a part of this project is presented from Chapter 4, where the procedure followed for acquisition and verification of data is presented. The effect of the chosen key and input data on the results of leakage detection methodologies is presented in chapter 5, the boards level variations and profiling templates is discussed in chapter 6. The properties of boards from both classes are compared in chapter 7, This report summarises the work and concludes in chapter 8.

2

Background

This chapter presents the background information forming the basis of the work conducted in this thesis. To explain data leaks, the implementation of a processor is discussed in section 2.1. Followed by discussion on cryptography in section 2.2 and present assembly implementation of an unsecured AES algorithm in section 2.2.2. Section 2.3 explains the link between assembly code execution by the processor and data leaks in power traces used for side-channel analysis. The chapter ends by discussing techniques for identifying leaks in section 2.4.

2.1. Microprocessor

Instruction set architecture (ISA) or hardware architecture refers to register transfer level (RTL) micro-architecture of core elements to provide the instruction set capabilities. We will be focusing on the ARMv6-M Thumb instruction set implemented on the Cortex-M0 core for our work.

The Cortex-M0 is a 32-bit RISC processor developed by ARM that implements version v6-M of the ARM instruction set [15]. It is one of the most widely used embedded devices due to an efficient instruction set, affordable development costs with comprehensive development tools and support. The Cortex-M0 has a Harvard architecture with both 16-bit (THUMB) and 32-bit instructions and a 32-bit data path. It does not include a data cache or memory management unit (MMU) but instead comes with a pre-fetch buffer. The ARM6 has 37 registers; consisting of thirty-one 32-bit general-purpose registers and six additional status registers.

2.1.1. Instruction set

The instruction set determines the functional capabilities of a processor by specifying the list of all instructions that can be performed. The basic instructions that can be supported by Cortex-M0 core are presented below with a brief description of it's implementation.

1. **Data processing instructions**- provide the functionality of manipulating with data, can be further divided into-
   (a) **Arithmetic instructions**- basic mathematical operations of addition, subtraction and comparisons along with logical operations (`EOR`, `AND`). These instructions are implemented on the ALU represented with green color in figure 2.2. The first operand is read from the register file using the A bus; the second operand can be sourced from either register file or memory using data-in register via B bus.
   (b) **Shift instructions**- Bit shift operations on data, logical and arithmetic. These are implemented using the barrel shifter represented with yellow color in figure 2.2.

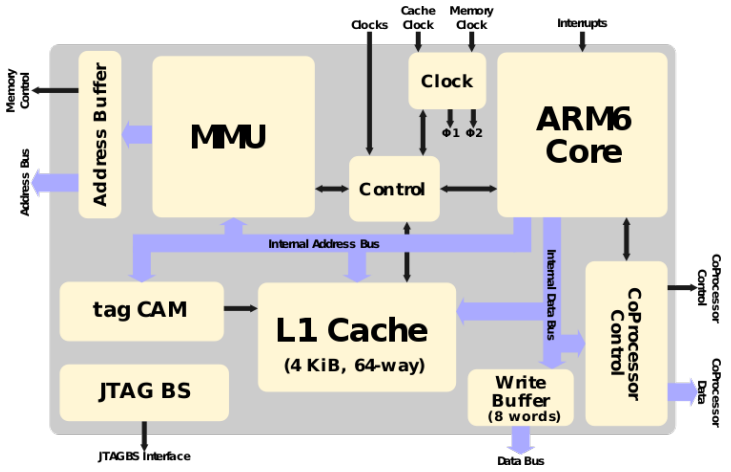


Figure 2.1: ARM6 hardware architecture [16]. The data busses are presented in purple and the control signals are presented in black. The micro-architecture of ARM6 Core has been presented in 2.2. It presents the different peripherals of an SoC and their interconnections.

(c) **Move instructions**-To move data between the register file represented with red in figure 2.2. Data moved across registers passes through ALU using A bus and is stored in the destination register from the ALU output bus.

Most implementations can compute the result of arithmetic and logical instructions in a clock cycle. Multiplication instruction is more complex to implement; the designers can decide between execution cycles needed to compute the result compromising area and power consumption. In Cortex-M0, the designers can choose the multiplier implementation amongst a 1 or 32 clock cycle implementation.

2. **Memory access instructions**- implement the read/write of data between registers and memory. Since these instructions access the memory, which takes time, it is normal for these instructions to take more than one clock cycle to execute. These instructions consume considerably more power in comparison to the arithmetic instruction.

   (a) **Load instruction**- Can read data from memory to register

   (b) **Store instructions**- Can write data from registers to memory

   Data loaded to the bus generated current in wires proportional to the hamming weight. Overwriting register values involves bit flips, which consume power relative to the old value on the bus. Hamming distance and hamming weight are the two widely used power models to profile the power consumption for predicting data.

3. **Branch and control instructions**- these instructions are used to control the execution flow of the program, enable code re-use and conditional execution. These instructions do not relate to data directly but rather allow controlling the address in the program counter, which determines the next instruction to be executed.

2.1.2. Toolchain

For executing code on a processor, the compiler and the assembler convert the code from a high-level language to a machine-readable form called **machine code**. The compiler performs the function of understanding the program flow for all different code sections and optimizes it for size and speed by removing redundant operations. The assembler breaks down the high-level code to perform the required operations by a combination of assembly instructions taken

from the instruction set of the architecture. The assembly code is translated to a machine code by replacing the assembly instruction with their respective **opcodes**; and a **binary** file is generated that can be decoded and executed by the processor.

The machine code is composed of sequential instructions that are executed by the processor core. Each instruction of the machine code is decoded by the processor to determine the operation to be performed. The instruction can also contain the source of operands and destination for returning the result based on the instruction functionality. The opcodes are used by the processor to select the corresponding micro architecture peripheral based on the instruction type, and it also configures the peripheral for executing the specific instruction by the generation of respective control signals.

The compiled and linked machine code is called a binary file because it consists of 1s and 0s, directly written to the program memory when the board is flashed. On booting, the system starts fetching the instructions and executing them from memory.

2.1.3. Micro-architecture

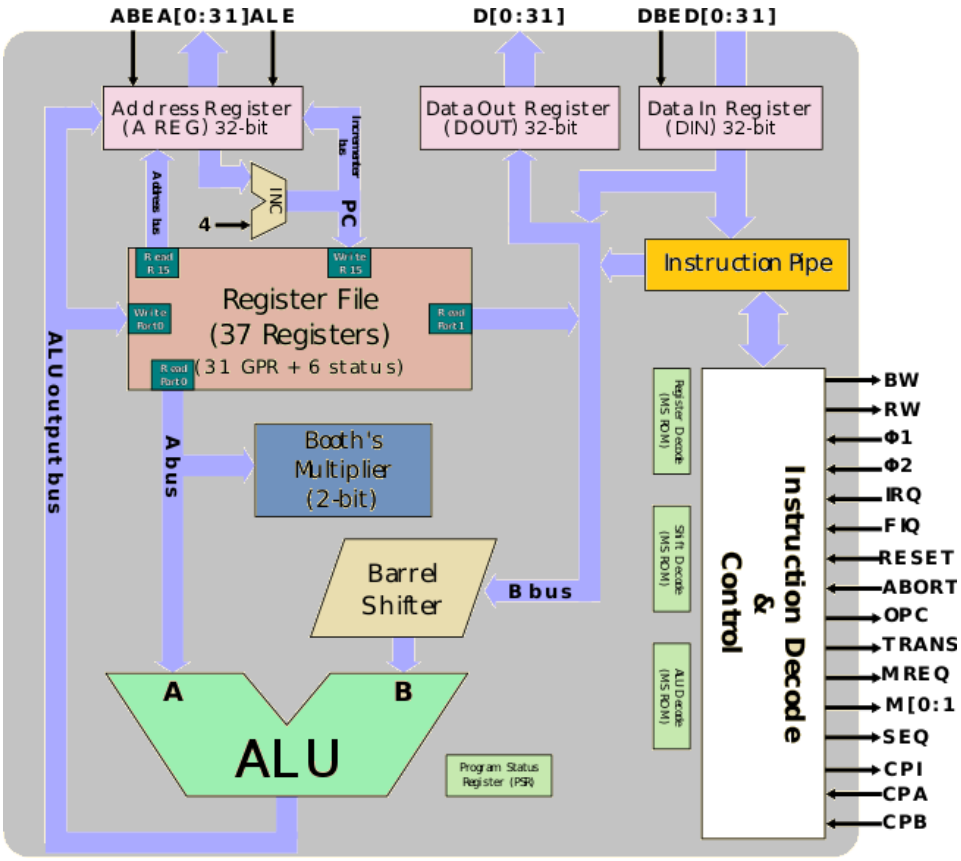


Figure 2.2: This figure shows the micro-architecture components needed for implementing arithmetic and memory instructions. The purple arrows depict the flow of data between the components. When an instruction uses secret-key data, it is loaded on the data bus to be used by the component performing the operation. internal control logic has not been represented in this figure; the black arrows represent the control signals applied to the core. Sourced from [16] and added color code with instruction labels

Micro-architecture refers to the specification and interconnection of different semiconductor peripherals, including registers(r0-r15), Arithmetic logical unit (ALU), multiply and accumulate (MAC) that provide the functionality to the core instruction set. Each micro-architecture peripheral is implemented by combinational logic gates to provide the operational functionality and routing circuits to transfer the data. One peripheral can support the execution of multiple

instructions; when an instruction is executed, the decoded opcode is used to select the respective micro-architecture element and activate it in the configuration required for performing the requested instruction.

There have been constant optimizations in design at the micro-architecture level for speed and utilizing the silicon area effectively; these optimizations can be in the form of multistage pipelines and resource sharing. The design specifications are mostly confidential as they are trade secrets. The device's functional specifications are provided to enable the developers to design applications without the need to know design specifications.

The arithmetic and logical instructions operate solely on registers. A barrel shifter located between the register file and the Arithmetic-Logic Unit (ALU) allows one to combine an ALU operation with a shift or rotation of the second operand. Most ALU instructions execute in one cycle; the only exceptions are MUL(multiply), DIV (divide), and operations targeting the program counter.

The core functionality of combinational logic gates and routing circuits is implemented by switching transistors. The switching of transistors gives rise to leakage current (the usage of term leakage here is in electronics terms) and electromagnetic emissions, which are recorded to perform side-channel analysis.

2.1.4. Subsystem implementation

This section will link how the instruction set and the micro-architecture implementation work to execute the machine code.

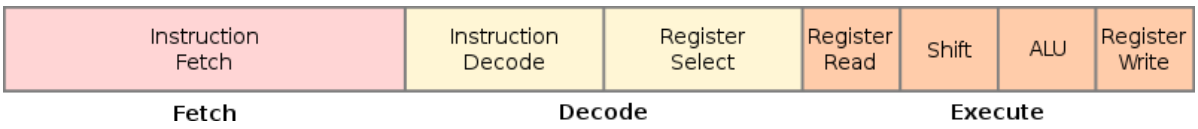


Figure 2.3: This figure shows the implementation of 3 stage pipline in ARM V6 core.

Cortex M0 has a three-stage pipeline, which implies three different instructions are being implemented in fetch, decode, and execute stages of the pipeline. To optimize the execution for speed, the memory operations are assigned to sub-peripherals to perform reads and writes, while the processor core can be executing other operations. Branch prediction is implemented where one of the alternative instructions is speculatively executed and eventually discarded if it turns out that the speculation was wrong. Store to memory instructions (e.g. str) are buffered and executed in one cycle, whereas load from memory instructions (e.g., LDR) introduce a wait-state.

Power trace shows the cumulative effect from all pipeline stages of the processor. Relating effects in power traces to instructions is problematic because the instruction being executed does not need to be the cause for leaks.

The memory location of instructions also plays a role because instruction pre-fetch works on preset word lengths. More than the requested data is cached to optimize for speed if the next instruction requires data in the current memory address's spatial vicinity.

Due to the reasons above, the methodology of side-channel analysis treats the target device as a black/grey-box and focuses on hiding/extracting sensitive data.

2.2. Cryptography

Cryptography has enabled the transfer of sensitive information while keeping it hidden from snooping third parties. Cryptography is the field of studying encryption/decryption protocols to implement secure communication by hiding the real content of data by using a secret-key. Though the secrecy of the message is dependent on the secrecy of the key.

Cryptography provides us with multiple cipher protocols for different use cases; the two broad classifications can be made as symmetric and asymmetric based on the nature of keys. Symmetric ciphers use the same secret key for encryption and decryption of data; AES is the current widely used block cipher introduced in 1998.

Our work focuses on recording power traces while the processor was executing AES encryption on 16-byte data. We discuss the methodology of AES in detail in the next section.

2.2.1. AES algorithm

AES stands for Advances encryption standard announced by National institute of standards and technology in 1998 [17]. AES is a block cipher whose key size can be amongst the three options of 128, 192, or 256. The applied key size determines the number of transformation rounds applied to the input data. AES encryption computes the cipher text from plain text and the secret key by using four significant operations on the 16-byte data represented in a 4x4 matrix.

1. **Key expansion**- The secret key is expanded in length from which sub-keys are of 128 bit are selected for each round following the key schedule
2. **Add round key**- 16 bytes of data are XORed with the 16 byte round key obtained from crucial expansion.
3. **Intermediate rounds**- In each of the intermediate rounds, the listed operations are performed. *Based on the chosen key size, the following operations are repeated 9, 11, or 13 times for key lengths of 128, 192, and 256 bits, respectively.*
   (a) **Substitute Byte**- The input bytes from the plain text are substituted based on a fixed table or s-box, and we have a 4x4 matrix with substituted values.
   (b) **Shift Rows**- The 4x4 matrix rows are shifted.
   (c) **Mix Columns**- This operation applies a mathematical transformation on a column of 4x4 matrix (4bytes) and outputs 4 bytes of data to replace the column.
   (d) **Add round key**- 16 bytes of data are XORed with the 16 byte round key obtained from key expansion.
4. **Final round**- Operations are the same as other rounds except for Mix Columns, we get the ciphertext as output from Add round key operation.
   (a) **Substitute Byte**
   (b) **Shift Rows**
   (c) **Add round key**

There can be multiple ways of implementing a specific function; for example, using look-up tables is the simplest means of S-box implementation, though it is unprotected from side-channel attacks. We discuss an unprotected AES implementation in the next section.

2.2.2. Assembly S-box implementation

In this section, the assembly code implementing the S-box operation used in our implementation is presented. This assembly code is generated by compiling Tiny-AES-C [1] library using Keil $\mu 5$ compiler. This is an unprotected AES implementation that is used to acquire power traces for our experiment. The background color of assembly instructions determines the type

[1] https://github.com/kokke/tiny-AES-c

of instruction as discussed in 2.1.1 and relates to the micro-architecture peripheral of similar color in figure 2.2.

The S-box implementation is unmasked and uses a look-up table stored in memory for performing substitutions. the relative addressing for the byte in operation is loaded in R3 with instruction `LDRB r3,[r3,r0]` address of look-up table is loaded in R4 with the instruction `LDR4,[pc,#28] ; @0x0800083C`, and finally the substituted S-box value is loaded to R3 with the instruction `LDRB r3,[r4,r3]`. The intermediate instructions of `LSLS` and `ADDS` are used to calculate the memory address where the result of the substituted byte will be saved by instruction `STRB r3,[r4,r0]`.

SubBytes	

(*state)[j][i] = getSBoxValue((*state)[j][i]);	LSLS r3,r1,#2
	ADDS r3,r3,r2
	LDRB r3,[r3,r0]
	LDR r4,[pc,#28] ; @0x0800083C
	LDRB r3,[r4,r3]
	LSLS r4,r1,#2
	ADDS r4,r4,r2
	STRB r3,[r4,r0]

Figure 2.4: `C` code implementing substitute byte operation (left); Assembly instructions sequence to implement the `C` code (right); **Colour code**: Yellow-Barrel shifter; Green-ALU; data-in/out registers-orange

To evaluate an application for security, we need to identify code sections that involve manipulation with sensitive data. This reduces the code section to be evaluated significantly; care should be taken to ensure the best coverage of the attack surface, which requires system knowledge. There have been attempts to remove this burden of security evaluation from the developer by designing leakage simulators to identify data leaks.

2.3. Side-channel analysis

Deducting information about the internal working of a system without interacting with it directly, rather observing on its interactions with the external environment is a side-channel. The side-channel signals are generated as a result of operating the device; they can be of different types such as power [5], electromagnetic emissions [18], or even sound [19]. Side-channel analysis involves processing these signals to deduct useful information that can be used to attack them.

As discussed in section 2.1, a processor provides its functionality by the combinational logic gates implemented at a micro-architecture level. Logic gates provide their functionality by switching transistors in the physical layer, which is done by generating a depletion region in the semiconductor gate. These processes at the semiconductor level occur due to electrons' movement in the circuit, which gives rise to leakage current and electromagnetic emissions.

We can record these side-channel signals from a device while it is performing operations by using a probe to convert the signal to a voltage domain that can be measured by an oscilloscope. The EM traces are recorded by positioning an EM probe over the silicon chip, which measures the EM field passing through the probe's aperture. To record power traces, we can add a tap on the device's power line in the form of a current probe or by measuring the voltage across a register. In power traces, we can observe the collective effect of all the micro-architectural components of the processor. In contrast, in EM analysis, it can be possible to perform localized measurements by selectively positioning the probe over the core. The work done in this thesis is has been applied to power traces; from here on, when we refer to trace, we mean power traces.

Side-channel can be distinguished into two sub-categories depending on the type of analysis performed. In **Simple power analysis (SPA)** the timing differences are used to make

deductions about the operations being performed, while in case of **Differential power analysis (DPA)** a amplitude deviations are compared from a collection of traces to comment on the operand data [12]. The power consumption of a device can be classified in terms of static power and dynamic power consumption. Static power refers to the constant power consumed by the chip circuitry, while the dynamic power consummation originates from the working of logic circuitry in the chip. DPA side-channel attacks focus on the dynamic power component because it has a strong input-dependency to accumulate information about a secret value manipulated by a device. Trace-sets with varying input data (IE-Fixed vs. Random) can be compared to distinguish between the static and dynamic power traces.

2.3.1. Simple power analysis

When a visual comparison of power traces is used to deduct information about the executed operation or branch, it is termed simple power analysis (SPA) [12]. We present a code snippet in figure 2.5 that executes *Addition* (`ADD`) , *Subtraction* (`SUB`) and *Multiplication* (`MUL`) instruction padded between `NOP` instructions to keep the processor processor and pipeline stages to be in idle state.

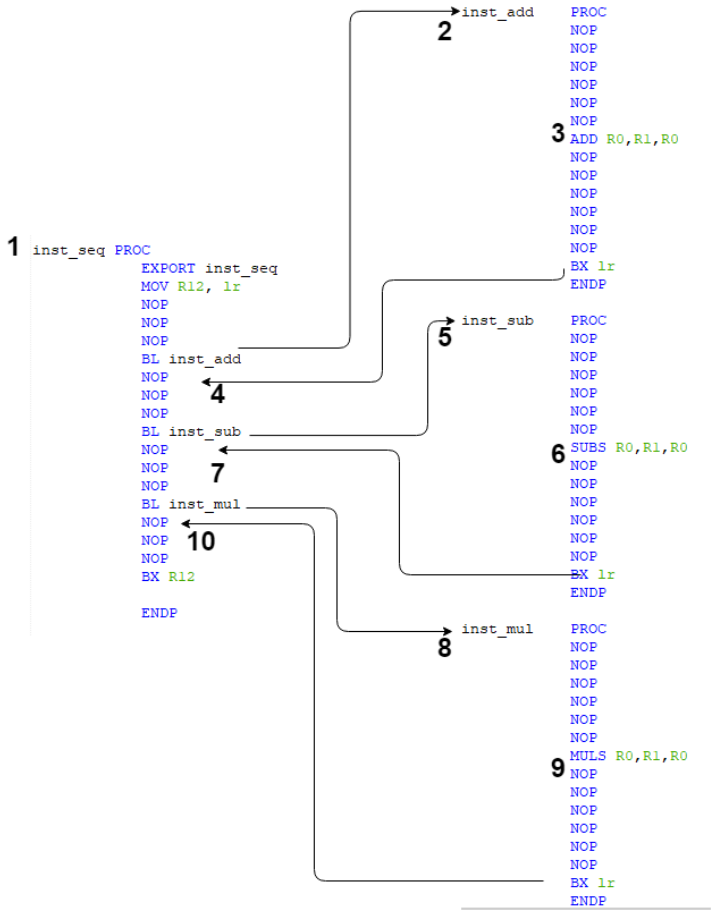


Figure 2.5: This figure shows a basic assembly code template that was created in order to to perform a visual inspection of how corresponding assembly instructions will be perceived in power traces. The label numbers in bold can be used to correlate the executed instruction to sections in power traces

A labeled power trace for executing a basic assemble code (shown in figure 2.5) on a STM32 is presented in figure 2.6. The control flow of program shown with arrows can be linked to different sections of power traces with the numbered labels. A the steep increases

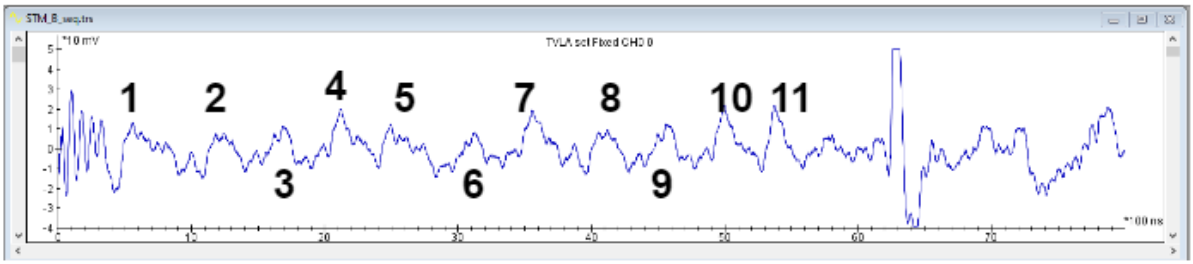


Figure 2.6: This figure the acquired power trace on executing the code snippet in figure 2.5 on a STM320f051 board. 1 marks the point where control jumps into `inst_sec` procedure

in current consumption on executing branching operations labeled by `1,2,4,5,7,8 & 10`. It is possible to distinct between *Branch to label* (`BL label`) instruction marked with `2,5 & 8` and *Branch to link register* (`BX lr`) instruction marked with `4,7 & 10` by the power traces of the instructions. In the same manner, it is possible to profile different instructions; the similarities in trace of arithmetic instructions labeled by `3,6 & 9` can be observed, pointing towards the use of same micro-architecture subsystems for executing the instructions. For easier comparison, we have extracted the sections of the power trace and presented it in figure 2.7.

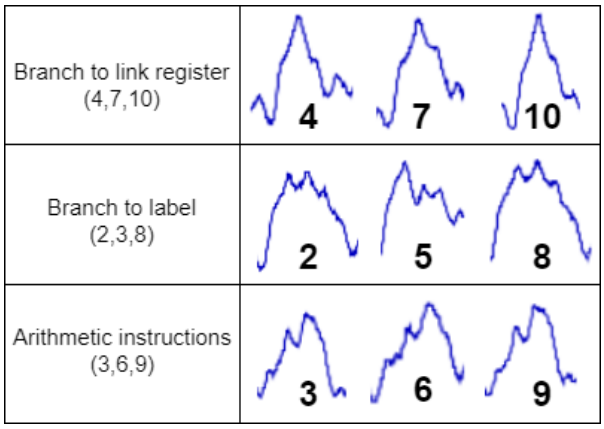


Figure 2.7: Signature of individual instructions shown in figure 2.6

By identifying the executed instructions from the power trace, an adversary can deduct information about the underlying implementation and use that knowledge to attack. Timings attacks are a subset of simple power analysis; they identify non-time constant executions and use them as a guessing oracle to observe how the execution changes with input [5].

Tanking an example of a password verification implementation that checks if the provided input is valid by iterating over input bytes sequentially. The number of iterations taken to compute the result needs to be balanced, or else an adversary can use the device as a guessing oracle. By providing different inputs for the first key byte, an attacker can compare traces to find the input that leads to the execution of a different code branch, implying the first byte passed the check. The above process can be applied iteratively for each password's byte.

Timing attacks are the most simple form of side-channel attacks; they can be mitigated by having constant time executions and balanced branches. We will now go over Differential power analysis, which can be applied in cases where simple power analysis cannot be used because of counter-measures used in the implementation.

2.3.2. Differential power analysis

When the branches are balanced and the intensity of leakage relative to noise is less, statistical tools are used on a large number of samples to deduct leaked information performing by Differential power analysis [12].

As discussed in the simple power analysis section, the instructions can be identified based on their power trace. We observe minor deviations in these power traces that originate from the instruction's data. These deviations are observed because reading and writing data from the data bus involves overwriting the old data with new data by toggling bits. These toggling of bits gives rise to leakage current whose effects we observe in instruction traces as small deviations. It is possible to profile the data from multiple traces with known data to create a profiling model that can be used to identify the underlying data from a power trace. By profiling trace-sets with mathematical and deep learning tools, we can accurately predict the secret-key data used to perform operations.

Due to various hardware cores available and the differences in the execution of software, it imparts too many variables to obtain a generic profiling methodology. The profiling has to be done specifically to the target device and software implementation for the reasons mentioned earlier. Moreover, two identical devices can significantly differ during manufacturing in the silicon layer, which induces significant differences between the two devices' power traces. Hence, for most cases, profiling has to be done on the same device, which is also the target. This has been a widely discussed topic in side-channel research to create machine learning networks capable of generating portable profiling models.

2.4. Leakage detection methodologies

Using input/output correlation is the most basic way to check for data leaks [20]. Over the years more advanced differential power analysis attacks have been developed exploiting different sources of information from the power trace. So statistical tools are used to find aberrations in the probability distribution function of trace-sets; the variation in probability distribution implies the source processes are different in some form and can provide information about the system.

The statistical tools can be used for different types of tests [21] The test can be classified as non-specific or specific, depending on the set of traces being compared. In the case of a non-specific test, a set of traces with a random inputs are compared against a set of traces with fixed input. In case of specific test, two sets of traces with the same fixed key but different fixed inputs are compared for differences in probability distribution function. If the probability distribution for the two sets, differs significantly the time samples can be classified as leaky [22].

The statistical tools of TVLA and KL divergence have been used in this work to check for leaks, are discussed in the sections ahead.

2.4.1. Test vector leakage assessment

TVLA methodology is a set of guidelines using Welch's T-test to check for a significant difference in the data set's true means [23].

T-test

A T-test is a statistical tool that tests if the null hypothesis specifying that the true mean of two distributions is zero. If the null hypothesis is proven false, it implies that the two distributions result from two distinct processes.

We calculate the **t-statistic** for two distribution P and Q, as follows

$$t = (mean(P) - mean(Q))/\sqrt{(variance(P)/N_P) + (variance(Q)/N_Q)}$$

where N_x is the number of samples in the set x.

This t value is compared with the threshold value, which is computed using the confidence level (denoted by α) of the test. If the t-statistic value is higher than the threshold, we reject the null hypothesis, implying the true mean of underlying distributions is not the same. Though the t-statistic is less than the threshold, we cannot say that the underlying distributions have the same true mean.

Confidence level and effect size

The confidence level is used to control the type-1 (false positives) and type-2 (true negative) error rates for rejecting the null hypothesis. It tells us if the difference of means between the two sets is big enough to suggest the true difference for the two data sets is non-zero. The threshold value for the T-test is based on the determined significance level α. When following the TVLA methodology, this threshold is set to 4.5 for a confidence level $\alpha = 0.00001$ [24].

Two sample populations are created at index locations being tested for leaks. If the result of the t-statistic is higher than the value of 4.5, we determine the index location be leaky. TVLA methodology emphasizes verifying the results by performing a repetition with second set of traces since it is a statistical tool.

2.4.2. Kullback–Leibler Divergence

Kullback–Leibler Divergence is an information theory metric that quantifies the difference between two probability distributions. It has traditionally been used in classifying the similarity in two sets. In the field of side-channel it has been used by J Park et al. for creating a side-channel instruction-level disassembler [25], which can generate assemble code from a single power trace. In our work we use it to quantify the level of dissimilarity in trace sets and contrast it with the sections of trace with leaks.

Let P and Q represent two probability distributions defined on the same probability space X, then KL Divergence of set P with respect to Q is calculated as

$$KL_{(P||Q)} = \sum_{x \in X} P(x) \lg(P(x)/Q(x))$$

KL divergence has a shortcoming that it is an asymmetric measure and cannot be used to measure the distance between two probability distributions.

$$KL_{(P||Q)} \neq KL_(Q||P)$$

The lowest value of KL-Divergence between two sets can be zero, which would imply both the distributions are identical.

$$KL_{(P||Q)} \geq 0$$

By using KL divergence to compare trace sets we quantify the differences in probability distribution function of trace-sets, to observe the behaviour for different instructions. By comparing the results of KL-divergence with leaks observed at that time sample, we comment on the viability of using KL-Divergence as a metric to test for leakages.

2.5. Key-rank analysis

To compare two implementations for security a unified methodology for the analysis of side-channel key recovery attacks is presented by FX-Standaert et al. [26]. Key-rank denotes the position of the correct key in the key guessing vector. Key guessing vector is a vector of all possible keys sorted from the most likely to the least likely. In our implementation operations are performed on each byte of data, so the key guessing vector consists of all 256 values of one byte. GE states the average number of key candidates an adversary needs to test to reveal the secret key after conducting a side-channel analysis.

Key-rank analysis is effected by the number of traces used for analysis. Key-rank evolves as more traces are added, if the correct key value is found to be leaking the key-rank converges towards 1. The key-rank evolution for 3 time sample from STM trace is shown in figure 2.8. The time sample plotted in red can be classified to have no leaks as the $log(keyrank)$ does not converges to 0. The time sample plotted in blue converges to 0 after 1000 traces, so the correct key does leaks. The time sample plotted in green shows high intensity of leaks as the correct key was found in less that 100 traces and it always stays at 0.

Figure 2.8: Rank evolution on STM device; plot for `LDRB r3,[r3,r0]` (red) with no leaks; `LDRB r3,[r4,r3]` (blue) with low intensity leaks, the correct key is found after 1k traces and `LSLS r4,r1,#2` (green) with high intensity of leaks, the correct key is found in less than 100 traces

2.6. Template attacks

Template attacks are a subset of differential power analysis as described in section 2.3.2, first works were performed by Chari et al. dating back to 2003 [27]. The principle of template attacks is to classify traces according to the leakage model and using Bayes's theorem to estimate the key data being used. The most widely used leakage models are hamming distance on 1 byte data which classifies into 9 classes or directly into 256 classes based on the byte value. Template consist of two phases, being profile phase and attack phase.

Profile phase involves classifying the traces data at the selected points of interest (POIs) to generate a leakage model. By using a known key value for the profiling phase, the intermediate value of operation can be computed and used to classify the traces. The number of POIs selection as well their location in time determines the quality of profiled template [28]. The

template is generated using probability distribution of different sample classes based on the leakage model. The closer the leakage model is to the actual power consumption the more effective is the attack.

During the attack phase the acquired power trace is compared against the template to characterize the intermediate value being computed upon. Sample from the attack traces are compared with the temple to determine the most probable key value. For a good attack it is necessary to build a template that has enough samples for each class of leakage model. Selection of leakage model influences the strength of attack, implying it will influence the number of traces needed to attack successfully [29].

3

Related Work

This chapter discusses the existing works, in the domain of side-channel analysis relating the effects of hardware to leaks. First, the works presenting micro-architecture sources of leaks are presented. Followed by a discussion on the portability of works preventing micro-architecture leaks.

3.1. Leaks from Micro-architecture

To attack a cryptographic implementation on a digital system, a good understanding of the system's underlying physics can provide an adversary with essential clues. At the same time, this knowledge is essential for a developer to assure the system's security. Hardware designers implement optimizations which can have unintended consequences leading to leaks from branch prediction [30] and instruction cache [31]. Data leaks can be classified based on their source as follows-

1. **Direct value leaks-** Operand value leaks because of current consumed by bus to transfer data
2. **Data-overwrite leaks-** Overwriting of data in register consumes power relating to the hamming distance of old and new value
3. **Circuit level leaks-** Originate due to coupling effects induced by current-carrying wires

Direct value leaks where the operand value leaks can be prevented by masking, though the system can still be exposed to data-overwrite and circuit-level leaks [32]. Y. Le Corre et al. discuss leaks due to pipeline stages [33] which is discussed in section 3.1.1; M. Renauld et al. investigates the significance of interconnects and coupling[34] discussed in section 3.1.2.

3.1.1. Pipeline leaks

Y. Le Corre et al. have presented A HDL level analysis of Cortex M3 core to explain the concepts used for building a Micro-Architecture power simulator (MAPS) [33]. Cortex M3 implements the ARM7 micro-architecture, the successor of the ARM6 micro-architecture used by Cortex M0. Thumb instructions used in crypto implementation use the basic functionality of micro-architecture peripherals, which is expected to be similar for ARMv6 and ARMv7; hence we use their findings to support our observations.

Y. Le Corre et al. analyzed the 3 stage pipeline implementation in ARM architecture and revealed the sources of leaks. They determine the presence of `ra` and `rb` register, located

between the register file and the ALU, being used to isolate the decode stage from the execution stage. For an instruction in the decode stage, the operand data is loaded into the co-responding registers; in the subsequent cycle to execute the instruction, the peripheral sources operand data from the intermediate `ra` and `rb` register. These intermediate registers combine the data from subsequent instructions, which is the source of data leakage. Loading of data to a register consumes power corresponding to the hamming distance [1] between the old value and new value, which leads to leaks on pipeline registers and register-reuse.

Operations at the hardware level involving the writing of a value to a register are a potential source of data-overwrite leaks. Y. Le Corre et al. determine the intermediate registers used as a boundary for the pipeline stages to be a source of data-overwrite leaks over subsequent instructions. They back their findings in the paper from experimental results on `ALU` and `STR` instructions.

3.1.2. Variability in nanoscale devices

M. Renauld et al., in their work [34] investigate the significance of variability in sub-micron technologies for cryptographic implementations. As the CMOS technology is scaled down, the static power consumption increases (energy consumed, even when no computation is performed), reducing dynamic power contribution. Device variability is becoming more significant due to manufacturing process variations making it challenging to determine a leakage model valid across physical board iterations. The tight integration of microelectronic circuits to minimize the area footprint increases the coupling between their interconnected parts; current-carrying wire can induce capacitance effects in its surroundings, leading to variations in path-delays. These path-delays are observed as **glitches** and leaks are termed **circuit-level leaks**.

To conclude devices have leaks from multiple micro-architecture sources; with the increase in process variability, different micro-architecture sources' contribution can vary across physical device iterations. Due to device-specific variations, the sample locations (from power trace) for generation/application of leakage models can vary. We test out the effects of device variability by generating and testing validity of template across different physical devices.

3.2. Template attacks and machine learning

Template attacks were first introduced by Chari et al. in 2002 [27], and since then these attacks have been recreated for breaking various hardware and software implementations [35, 36]. Template based DPA attacks have been considered one of the most powerful attacks from information theoretic point of view and have shown success in breaking masked implementations also [37]. Earlier works in template attacks used the same physical device for profiling phase and attack phase, now there is a move towards working on portability of these attacks.

The major problems in porting template attacks arises from the fact that side-channel traces have noise, being influenced by environmental factors (temperate, power source) during acquisition which effect the acquired traces. The second more prominent problem arises from the presence of inter-device variations, which make it difficult to have a standard leakage model for a class of target device. The problem of portability has been approached from different directions, by using algorithms to normalise the drift in acquisitions [38]. The work done by X-DeepSCA: Cross-Device Deep Learning Side-Channel Attack [39] has been successful in creating a model that can be ported across devices, by using traces from multiple devices. Work done by S. Bhasin et al. has evaluated the portability for profiled side channel attacks and has shown the advantage of using multiple device models for portability of attacks [40].

[1] number of positions at which the corresponding bits are different

The strength of attack is greatly influenced by the quality of profiled template; the number of features (POIs) used for creating template, and the time samples selected as POI and the leakage model used have a big impact in the quality of template. The work done by S. Picek et al. has investigated the use of machine learning with different tuning parameters and have compared it with the conventional template attacks [41]. Their work shows machine learning can outperform the conventional template attacks when applied properly.

3.3. Portability of works preventing micro-architecture leaks

Developing a secure software implementation involves determining the cause of leakage and fixing it until no more leaks are found. Advanced tools for simulating and detecting leaks from hardware are presented in this section. These tools provide a good means to test for data leaks, though their limitation which have prevented these their wide use are also presented.

Clearing the registers before loading the subsequent value can prevent data-overwrite leaks; this applies to both register and intermediate registers [42]. M. Renauld et al. have developed Micro-Architectural Power Simulator (MAPS) [34], which takes the generated binary and simulates the data flow through all the registers in the pipeline to predict data-overwrite leaks. To add support for another target, analysis of HDL schematic is required, this information is not always available.

There is no easy way to predict circuit-level leaks as the circuit delays influence them at the gate level and induced capacitance effects from wires. The effects of circuit delays can be simulated on the RTL schematic but are computationally complex, and RTL schematic is not always available. It will still not include the leaks due to capacitance effects. D. Mccann et al. have developed ELMO by creating leakage models of instruction triplets (Cortex M0 has 3 stage pipeline) for basic crypto instructions [13]. Their methodology encapsulates detection of circuit-level and data-overwrite leaks, but adding support for new targets is labour intensive. The complexity of modeling increases with the number of pipeline stages in the target processor. There are further works that replace leaky assembly instructions with a sequence of assembly instructions providing the same functionality at the cost of executing additional instructions [43, 44].

Based on the type of leaks, appropriate remedial action can be applied. Understanding the data-flow at the micro-architecture level and clearing the register value before subsequent writes may prevent data-over write leaks. Circuit level leaks are more difficult to prevent as specific requirements need to be laid down at the RTL design stage to prevent data leaks through glitches or be mitigated at the software level using alternate instruction sequences. Solutions have been proposed at the hardware level to develop augmented instruction set architecture (aISA) specific to secure operations. However, it needs to be incorporated into the manufacturer's design and it costs area and design overheads [45].

RISC hardware architecture provides a separation between the instruction set and its micro-architecture implementation, providing manufacturers' flexibility for designing cores optimized for specific requirements. Having the same instruction set allows portability of application across devices, though differences in micro-architecture can impact the implementation's security. This presents the problem in the portability of crypto-implementations across devices with the same instruction set since the hardware implementation of instructions at RTL level vary; so the countermeasures used have to be evaluated specific to the device.

4

Experimental Setup

In this chapter, we go over the two classes of boards use for our experiments in section 4.1, followed by the acquisition setup used for collecting power traces from the target boards in section 4.2, We discuss how to check if the raw traces are suitable for performing comparisons in section 4.4 and end the chapter by explaining the procedure for making the traces from two different classes of board comparable in section 4.5.

4.1. Target devices

Two devices designed on `Cortex M0` core were selected for this study because comprehensive literature is available and Cortex M0 is the most basic 32-bit processor core, with a wide application in embedded and IoT devices, making it the right candidate for our work. The future generations of ARM cores have been designed around the Cortex M0.

Embedded devices are designed around a processor core by semiconductor manufacturers to create application-specific systems on chip (SoC). The distinction between a processor core and an embedded device is essential. When the term device is used, it implies the specific implementation of a processor core, the features of the device are dependent on the manufacturer's design choices.

Our work explores the impact of hardware architecture on leaks observed from power traces. The target devices devices from different manufactures have different design specifications but are built around the same hardware architecture (processor core). Such comparison can provide us with information for relating data leakages to the micro-architecture effects in physical layer.

The target devices chosen by us for our experiments are-

1. STM32F051: We used two STM32 boards for our experiments, we refer to them as `STM_A` and `STM_B`.

2. NRf51: We used two NRF boards for our experiments, we refer to them as `NRF_A` and `NRF_B`.

Signal to noise ratio in the acquired traces significantly impact the results of side-channel analysis. Leakage detection methodologies are discussed in 2.4 are dependant on the probability distribution function of the leaks. The power traces should have a high signal to noise ratio to get good results from the leakage detection methodologies.

Based on the target device and the acquired traces, getting a high signal to noise ratio might require reduction of interference from the board circuit circuitry. Hardware modification

can be done on the board circuitry after studying the schematic for the same. Care has to be taken when performing physical modifications, not to impact the performance or execution of the processor core.

In the following section, we present the specifications devices from STM class (4.1.1) and NRF class (4.1.2) along with the hardware modifications made before moving on to acquiring traces.

4.1.1. STM32F0 Discovery

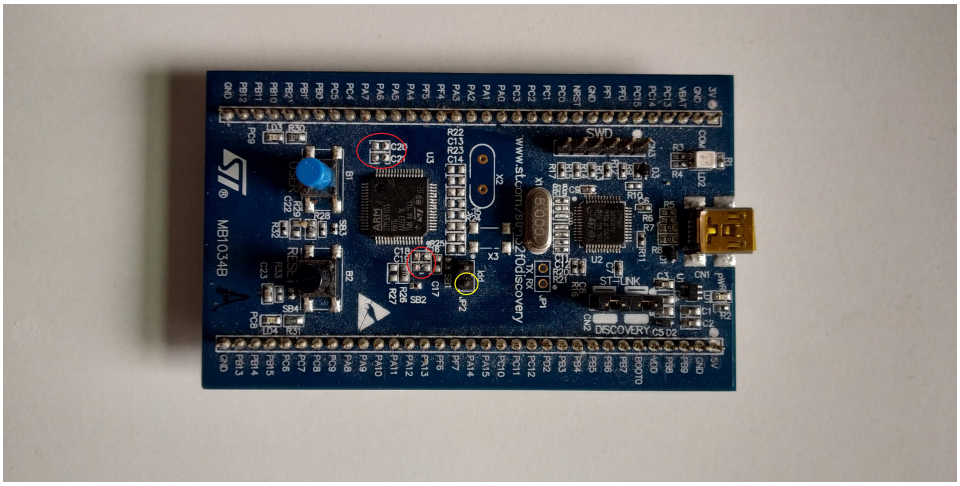


Figure 4.1: This figure shows the STM32F051 development board along with the hardware modifications, used to collect traces on AES encryption. The coupling capacitors `C18, C19, C20 and C21`, highlighted by a red circle in the image, were removed. To isolate the target MCU from the effects of interface MCU, it was powered from the current measurement port by the pin highlighted by the yellow circle. The target is powered by a 3.3V supply with the current probe attached in series. The interface MCU is powered by the USB.

STM32 Discovery is a development board by ST Microelectronics for the STM320f051 device, which consists of an interface MCU on-board that enables easy flashing and debugging using ST-Link over USB. The development board also offers a PPI port that connects a current probe in series to measure the current consumption. The hardware modifications made on the device will now be discussed.

On inspection of the STM board's schematic, we observe that interface MCU and the target device share the same power source. Even though the PPI port is used to measure the target MCU's current, by sharing the same power source for both MCUs, the power line becomes unstable, resulting in more noise in the acquired signal. The target MCU is powered by an external 3V3 supply from the current measurement port while having a current probe in series, while the interface MCU is powered using the USB port. This way, we isolate the target MCU on the power feed from the effects of interface MCU, which is now powered by USB.

The coupling capacitors attached to the power pins of target MCU are removed; they act as a low pass filter on the input power supply to target MCU. Filtering can lead to loss of information in the power traces at the cost of low-frequency noise in the signal. The coupling `C18, C19, C20 and C21` are removed, as shown in figure 4.1 without affecting the device's performance.

4.1.2. NRf51 Development kit

NRf is a SoC designed for Bluetooth low energy application based on `Cortex M0` running at 16MHz. We choose this device to have a comparison for STM device since NRf51 is a low power device using a fraction of the power of the STM board even though both the devices are

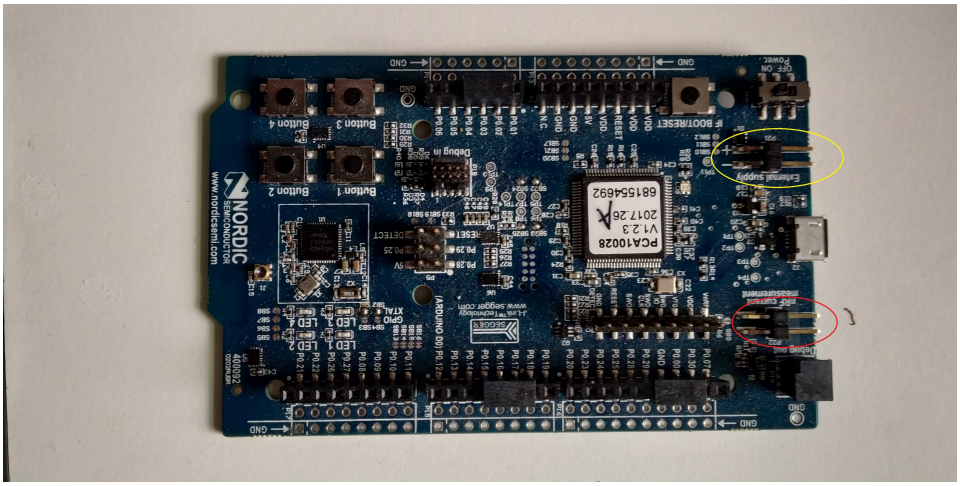


Figure 4.2: This figure shows the NRF51 Development kit 1 board on which AES implementation was run. The board was powered from the external power supply port highlighted with a yellow circle by a 3.3V supply. We did not have to make hardware modifications for this board as the interface MCU is isolated by default on powering the board from an external supply. The current probe was attached to the current measurement port highlighted by a red circle in the image

designed on the `Cortex M0`. NRF51 development kit also offers a current measurement port that can be enabled by cutting solder bridge `SB9`. No coupling capacitors were found on the power-line circuitry. The target MCU is already isolated from the interface MCU when the board is powered externally using 3v3, so we do not need to make any hardware modifications.

Since NRF51 is a low power electronic device, Picoscope 3000 has the lowest resolution of 39 µAAmps, but it cannot record good traces due to quantization noise. The current measurement signal is passed via a signal amplifier to improve signal gain, as presented in figure 4.4.

4.2. Acquisition Setup

The prepared boards are now ready to acquire the traces. We connect the current probe to the PPI port on the board. The PPI port is in series to the target MCU's power line and can be used for taping into the current consumption of the same. Using a current probe induces a phase shift; hence we add an inductance of 50 OHMs before the signal is recorded from the oscilloscope. Riscure Inspector [1] is used to acquire the power traces; the acquisition setup used are presented in the image. The trigger pin on the board is connected to the channel B on the oscilloscope, which starts recording measurements from channel A on detecting trigger signal. We use a USB UART cable for connecting to the UART pins on the board from USB, the serial communication to the device happens over this cable.

The specific pins used for STM32 and NRf are shown in figures 4.3 and 4.4 respectively. The specifications of the tools used have been presented below.

Acquisition software

Riscure Inspector Version: 2019.2.1 Riscure inspector is a side-channel analysis tool that enables acquiring traces from target, while combining it with input (plain text) and the output (cipher text) from target device. It incorporates a signal processing toolkit to apply pre-processing on the acquired traces along with modules specific to SCA (i.e. TVLA, key-rank analysis, template attacks).

[1] https://www.riscure.com/security-tools/inspector-sca/

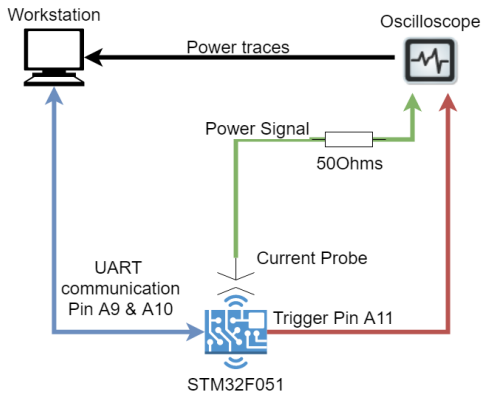


Figure 4.3: Acquisition setup used for STM board. Pin A9 and A10 are used for UART Rx and Tx respectively, pin A11 is used for connecting the trigger signal. Current probe is attached to the oscilloscope though a 50 Ω impedance

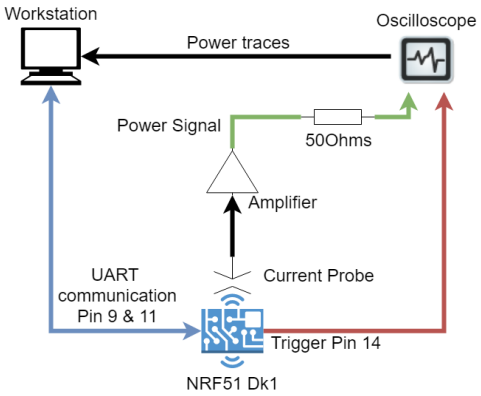


Figure 4.4: Acquisition setup used for NRF board.Acquisition setup used for STM board. Pin 9 and 11 are used for UART Rx and Tx respectively, pin 14 is used for connecting the trigger signal. A signal from the current probe is passed through a signal amplifier because NRF51 is a LPE device, in order to increase the gain of the signal. The power signal is fed to the oscilloscope via 50 Ω impedance

Oscilloscope

Model: Picoscope 3000 **Sampling rate:** 500MHz **Range:** $\pm 50mV$ **Resolution:** 8 bit

Current Probe

Model: Risc-CP189

4.3. Preparing portable software implementation

Our work is focused on comparing power traces across different target boards, so both the boards must be executing the same instructions in sync. Since the boards from different manufacturers will have a different project for configuring and running the startup script, we have to make sure the region of code between the triggers is identical for both targets. The same compiler was used to generate the binary files, and the disassembly of code was compared to ensure that the execution would be the same.

An unmasked implementation of AES128 was flashed on both target boards. The execution sequence is explained below.

1. On boot/reset, a startup code runs on both target devices, which sets the system and peripheral clocks. This step differs for two boards as follows
   - **NRf51:** works at a fixed clock speed of 16Mhz.
   - **STM:** device supports operation over a wide clock frequency. The startup code sets the clock frequency to 16MHz.
2. Core and UART drivers are initialised
3. System tick interrupt is disabled
4. Control enters `main()` and an AES object is initialised with a preset key
5. Enter infinite loop, repeat the steps below
   (a) Receive 16 bytes of data over UART.

(b) Set **trigger pin low**. this signals the oscilloscope to start recording the measurement.

(c) 16 bytes of data is encrypted using an unmasked implementation of MBED-AES

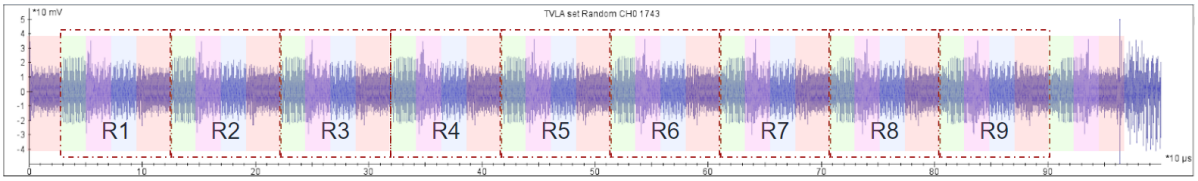
(d) Encrypted ciphertext is sent back over UART

By setting the clock frequency of both the board to 16MHz, both the target devices can be configured for identical execution. Identical execution timings are achieved at the clock cycle level for the uploaded code, which is an important criterion for comparing power traces across different target devices. Only the required peripheral drivers of UART were initialized to keep the effects from core peripherals at a minimum in the recorded power traces. The system tick interrupt was disabled to prevent switching of program control in the middle of AES encryption operation since it would lead to inconsistencies in execution traces.

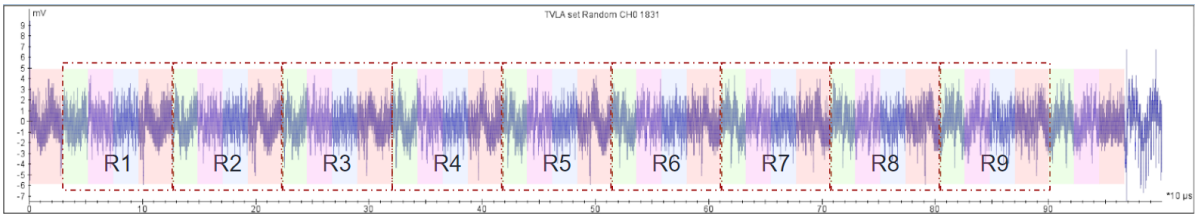
On successfully receiving 16-bytes of data, the trigger pin is set to high, which signals the oscilloscope to start recording the power measurements. After setting the trigger pin high, program control enters the `AES_ECB_encrypt` function, which computes and returns the ciphertext as output, which is sent back to the workstation over UART. The software projects for both the devices were created and compiled using Kiel μVision5 to have a consistent development tool-chain.

The AES library used is an insecure implementation from MBEDtls.

4.4. Acquired traces



(a) Raw power trace acquired from STM board for AES-128 implementation.



(b) Raw power trace acquired from NRF board for AES-128 implementation.

(c) Legend for the colour overlays to represent the operation being executed.

Figure 4.5: We can observe how the execution of operations is synchronised between the two devices even though the traces from both boards are not identical.

The raw traces acquired from STM and NRf board using Riscure Inspector are presented in figure 4.5. From a visual comparison, it is evident that the power consumption across the target devices differs significantly. It was expected since different manufacturers make devices with different specifications. Moreover, NRf51 is an LPE device, and the power traces were acquired using an amplifier.

The different operations being performed based on repetitive patterns can be distinctly identified across both the target devices. The raw traces have a color overlay depicting the AES operations being performed at the trace sections, corresponding to the sequence of operations

discussed in section 2.2.1. The operation execution across both the devices is synchronized in time, which is in line with our expectations to ensured identical execution of code across both devices.

4.4.1. Organisation of datasets

The power traces coming from different devices of the same Cortex-M0 family differ significantly visually. Detailed analysis is performed on trace-sets consisting of multiple traces. Each trace-set consists of 2.5k encryption traces using the same secret key. Half of the trace-sets are provided with a 16 byte fixed plain-text and the collection of these traces is termed as fixed-set. The other half is provided a 16-byte random plain-text; the collection of these traces is termed as random-set. A 2.5k trace trace-set is a union of 1.25k trace random-set and 1.25k trace fixed-set. Each trace has the input plain-text and output ciphertext information embedded in them.

We collected multiple traces-sets from the target boards executing AES encryption varying the parameters given below-

1. **Board family**: The label `STM` or `NRF` determines the family of board from which trace-set was acquired.

2. **Board iteration**: We have two development boards from each family labeled as `A or B`

3. **Encryption key**: The secret key used for encryption labeled by `key1` or `key2`.

4. **Fixed plain-text**: Determines the 16 byte fixed input provided to half of the traces represented by `D1` or `D2`.

5. **Repetition**: TVLA methodology specifies performing a repetition in order to verify results, trace-sets are labeled by `1` or `2` representing two repetitions

We choose a naming convention for easier identification of trace-set parameters post-acquisition. The nomenclature followed is `class_board_key_data_repetition`, for example a trace-set with the name `NRF_B_2_a` means it was collected from `NRF` board `B` with `key 2` and provided fixed input `a`. In total, we collected 32 traces sets with different permutations of parameters, all of which have been listed in figure 4.6. We will be referencing these trace labels to identify the trace-sets and their parameters while discussing data analysis.

	STM Board			NRF Board	
	Fixed set-1	Fixed set-2		Fixed set-1	Fixed set-2
Board A	STM_A_key1_D1_1 STM_A_key1_D1_2	STM_A_key1_D2_1 STM_A_key1_D2_2	Key 1	NRF_A_key1_D1_1 NRF_A_key1_D1_2	NRF_A_key1_D2_1 NRF_A_key1_D2_2
	STM_A_key2_D1_1 STM_A_key2_D1_2	STM_A_key2_D2_1 STM_A_key2_D2_2	Key 2	NRF_A_key2_D1_1 NRF_A_key2_D1_2	NRF_A_key2_D2_1 NRF_A_key2_D2_2
Board B	STM_B_key1_D1_1 STM_B_key1_D1_2	STM_B_key1_D2_1 STM_B_key1_D2_2	Key 1	NRF_B_key1_D1_1 NRF_B_key1_D1_2	NRF_A_key1_D2_1 NRF_A_key1_D2_2
	STM_B_key2_D1_1 STM_B_key2_D1_2	STM_B_key2_D2_1 STM_B_key2_D2_2	Key 2	NRF_B_key2_D1_1 NRF_B_key2_D1_2	NRF_B_key2_D2_1 NRF_B_key2_D2_2

Figure 4.6: Created a data set with 32 trace-sets following the defined nomenclature showing set parameters.

In the next section, we will be discussing the methodology used for the synchronization of traces and the addition of instruction labels to the traces.

4.5. Preparing traces for comparison

Even though the boards execute the same instructions, the traces must be synchronized after the trigger finely align the clocks. To perform an instruction level analysis of traces, cycle-accurate instruction labels are added in the time axis. We will now discuss how these problems were solved.

4.5.1. Synchronisation of traces

Having the trace-sets aligned finely in time along clock cycles is necessary to provide an accurate analysis of the observed effects. The accuracy of the trigger signal determines the level of drift that is observed between the acquired traces. We compared the accuracy of the trigger signal from the oscilloscope to the recorded traces to find the level of drift.

Using the disassembly of C code, the assembly code line that sets the trigger pin low is found to have exact timings. The number of cycles the program flow takes to enter the `AES_Encrypt` function is noted, which is used as an offset to identify the start of `AES_Encrypt` in recorded traces. The resulting offset from above was applied to the boards' traces to get them synchronized.

The synchronization of traces is verified by applying correlation between the amplitude of power trace with the input bytes. Doing so reveals the section of traces influenced by the respective input byte data. The results of the correlation are presented in figure 4.7 for both the device classes. The results show similar shape for both the target devices, which is a good sign showing that the similar section of trace is influenced by the input data. The shape of curves are aligned in time for both the boards, this proves the success in alignment of traces.

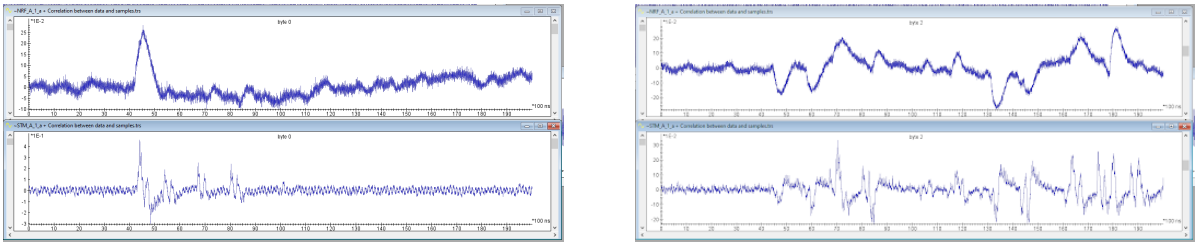


Figure 4.7: Correlation results for STM (bottom) and NRF (top) boards

4.5.2. Adding instruction labels

Arm process simulator in Keil MDV version 5 was used to record the execution trace of instructions inside the `AES_encrypt` function. Execution trace information from the simulator is used to link the instruction labels with their power traces segments. It outputs a CSV file with disassembly code along with the timing of execution of instruction. On saving the execution trace, we get the instruction labels along with their cycle-accurate timings, as presented in image 4.8.

The Instruction trace CSV was used to add instruction labels on the X-axis; the timing information was used to determine the segment of power trace over which the instruction is executed. Using the above process, we can isolate trace segments specific to instruction for analysis.

The results of combining power traces to instruction labels has been presented in image 4.9 and 4.10. which presents the acquired power trace immediately after acquisition up until `add_round_key` operation on the first four bytes.

In the raw traces acquired from STM and NRF board, repeating sequences of instructions show a similar trend in power traces. This lets us conclude that the traces are synchronized correctly with the instruction labels.

	A	B	C	D	E	F	G	H	I
1	0	0	0.000454	0x000007B0	6088	STR r0,[r1,#0x08]		main	
2	1	125	0.000455	0x000007B2	A930	ADD r1,sp,#0xC0		main	
3	2	187	0.000455	0x000007B4	4668	MOV r0,sp		main	
4	3	250	0.000455	0x000007B6	F7FF	BL.W __semihosting_library_function (0x0000025A)		main	
5	4	437	0.000455	0x0000025A	B570	PUSH {r4-r6,lr}	{	AES_ECB_encrypt	
6	5	750	0.000455	0x0000025C	4604	MOV r4,r0		AES_ECB_encrypt	
7	6	813	0.000455	0x0000025E	460D	MOV r5,r1		AES_ECB_encrypt	
8	7	875	0.000455	0x00000260	4621	MOV r1,r4	Cipher((	AES_ECB_encrypt	
9	8	937	0.000455	0x00000262	4628	MOV r0,r5		AES_ECB_encrypt	
10	9	1000	0.000455	0x00000264	F000	BL.W Cipher (0x000002B2)		AES_ECB_encrypt	
11	10	1188	0.000456	0x000002B2	B570	PUSH {r4-r6,lr}	{	Cipher	
12	11	1500	0.000456	0x000002B4	4605	MOV r5,r0		Cipher	

Figure 4.8: Trace execution data showing cycle accurate timings, machine code and their assembly instructions

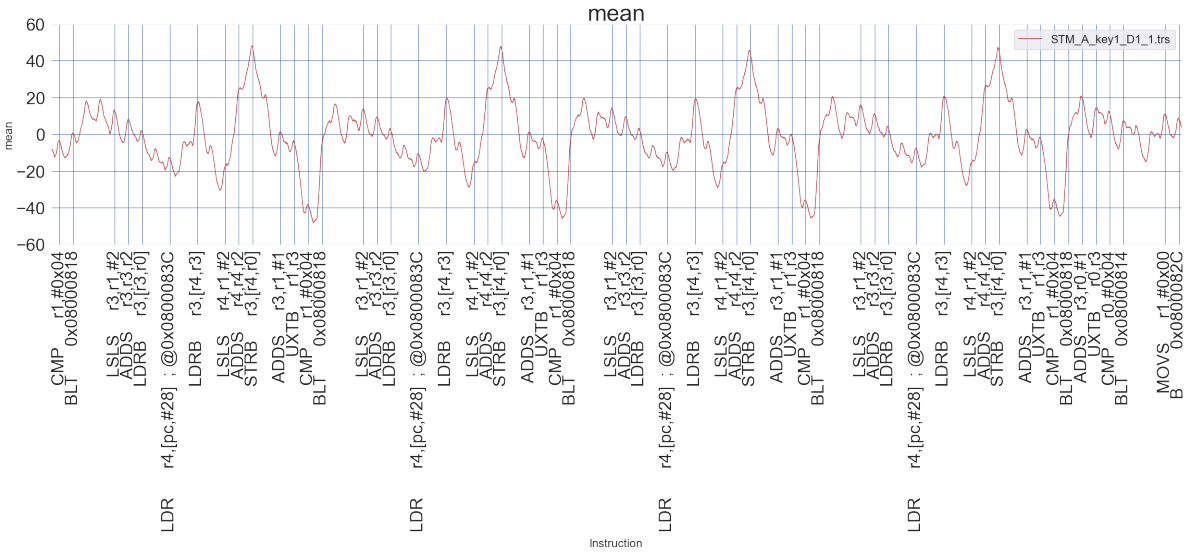


Figure 4.9: Power trace of 1st round S-Box operations with instruction labels for the STM board, repetitive shape of power trace for instructions proves accurate synchronisation of instruction labels

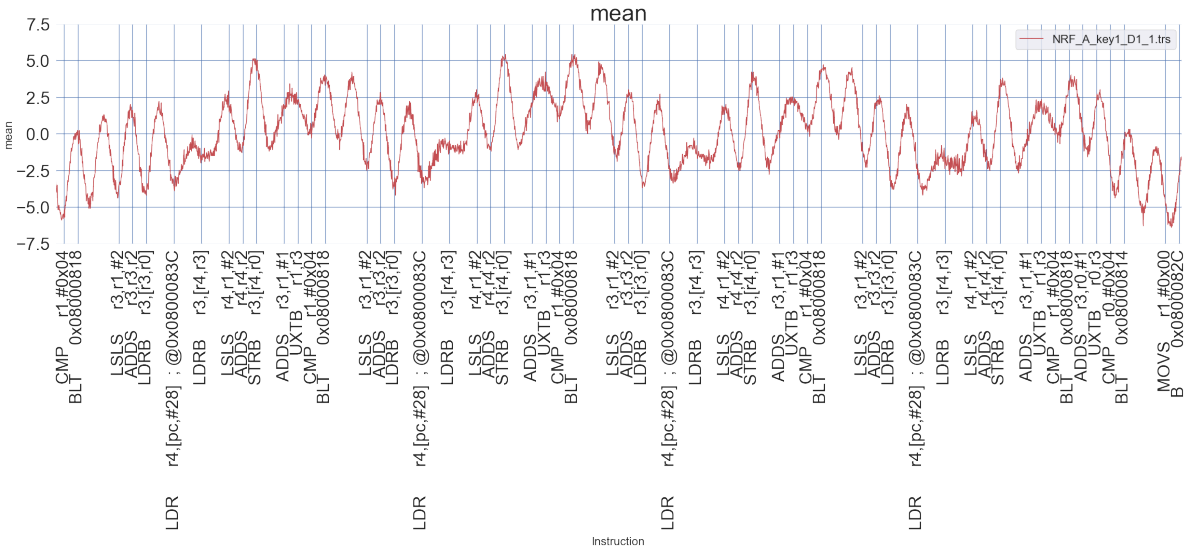


Figure 4.10: Power trace of 1st round S-Box operations power plot with instruction labels for the NRF board, repetitive shape of power trace for instructions proves accurate synchronisation of instruction labels

In the next chapter, we discuss the methodology we used for analyzing these data-sets.

5

Intra-board comparison

In this chapter, we answer the research question, how significant are the inter-key and inter-data variations? To answer this question trace-sets are acquired from a physical board varying the input data keeping key constant for inter-data sets and changing the encryption key keeping input data constant for inter-key trace sets. The experiments are performed at the board level, where traces from the same physical boards are used for comparisons. A power profile comparison based on the mean and standard deviation is performed, after which leakage detection methodologies are used to check for the leak of correct key data.

The methodology used for comparisons for comparison is explained in the first section 5.1. After which comparison of STM class boards is presented 5.2, followed by the NRF class boards 5.3; we conclude the chapter by presenting the observed properties across devices of both class 5.4.

5.1. Methodology

The purpose of is to compare trace-sets collected from different physical boards. varying in parameters. Each trace-set is a union of a **fixed-set** and a **random-set**; the fixed-set is composed of 1.25 traces with a fixed input, whereas the random-set is composed of 1.25k traces with random input. Each trace index is analyzed using **sample-sets**, created from the collection of samples at the specific index from all the traces in trace-set.

A comparison is performed by applying the selected measure on sample-sets along the trace indexes to be analyzed. The resulting value of applying measures on the sample-set is plotted at the coresponding index to obtain a **measure-plot**. The measure-plots are labeled by coresponding parent trace-set. We compare the similarities and deviations in the measure plots from different trace-sets to comment on the results.

The result of a repetition trace-sets is also computed to verify the observations are not incidental and can be repeated, to report results with confidence. Solid and dotted lines distinguish the two repetitions.

To answer the question of how similar are two trace-sets, we perform a power profile analysis, which has been expanded in section 5.1.1. For answering the question we perform a comparison based on data leaks, which will be discussed in detail in section 5.1.2. Known-key analysis is performed to comment on leak of correct key data.

5.1.1. Power profile analysis

A power profile analysis is performed to determine the differences in the power traces across executions. By comparing the statistical properties of trace-sets, we can determine the deter-

mine how the underlying probability distribution is influenced by a change of parameter. The resulting measure plots are observed for similarities and dissimilarities to make conclusions. In Power profile analysis, comparisons are made between measure-plots of random-set and fixed sets, (non-specific tests discussed in 2.4). The influence of input data on the probability distribution of samples is being checked across fixed-set versus random-set. Operations dependent on the input data can be determined based on the result of this comparison.

5.1.2. Data leakage analysis

Data leakage analysis is performed by applying leakage detection methodologies on the sample-sets. The measures of leakage detection compute results based on underlying distributions for the two sample-sets. In our methodology, we are performing non-specific tests consisting of a fixed-set and a random-set of traces.

The current state of the art **TVLA methodology** (discussed in 2.4.1) is used along with a newly proposed **KL-Divergence** (discussed in 2.4.2) to check for data leaks. Though both these measures predict leaks along the power trace, it might not necessarily represent leak of correct key data. Key-rank analysis is performed to determine the probability of correct key data being leaked to an adversary.

5.1.3. Known key analysis

The known-key analysis module from Riscure Inspector software is used to perform key-rank analysis. In known-key analysis, a random trace-set is used to profile the traces using the selected power model, in our case hamming weight model was used. The profiling stage uses the input data to categorizes traces depending on the power model. In a hamming weight model, we have nine different classes. The input data is used to compute the hamming weight for classifying the trace, which is correlated with the input data.

The profiled model is then used to predict the leaks along the trace by ranking the probability of leak for all the possible key-byte combinations. The trace sections is leaking key-data when the correct key is ranked highly. If the correct key is ranked one, it has a very high probability of leaking to an attacker. The number of traces in the random-set also influences the results of the key-rank analysis. The more traces we have for the profiling stage, the better the model and prediction are. The profiling quality is determined by the number of traces being used, though it saturates after a limit. Known key analysis relates observable leaks at each index and determines the probability of correct key byte leaking to an attacker.

S-box key-ranking results

The result of applying known-key analysis on the STM board is presented in figure 5.1. The sequential operation on all 16 bytes is shown, overlayed by the intensity of leak for each key byte. We can observe the s-box implementation operating on data in groups of 4 bytes; the process is repeated four times.

In the STM board, the key data leaks strongly while the subsequent byte is loaded and operated upon it. The value leaks when the next key byte data is being loaded and computed upon, this can be an evidence for data-overwrite leaks from registers. A small section of leaks is observed again when a key element from the same group is being operated upon. This can relate to how key-data is stored in subsequent memory locations, and memory access loads more than 1-byte data on the bus. This effect can be due to 4-byte memory access in Cortex M0; the old key bytes are also sent on the bus due to word size of 4 bytes. (i.e. in figure 5.1 We observe leak of k[0][0] when operations are performed on k[0][1]. We observe leak of k[0][0], k[0][1] when operation are performed on k[0][2]; similarly We observe leak of k[0][0], k[0][1], k[0][2] when operation are performed on k[0][3].

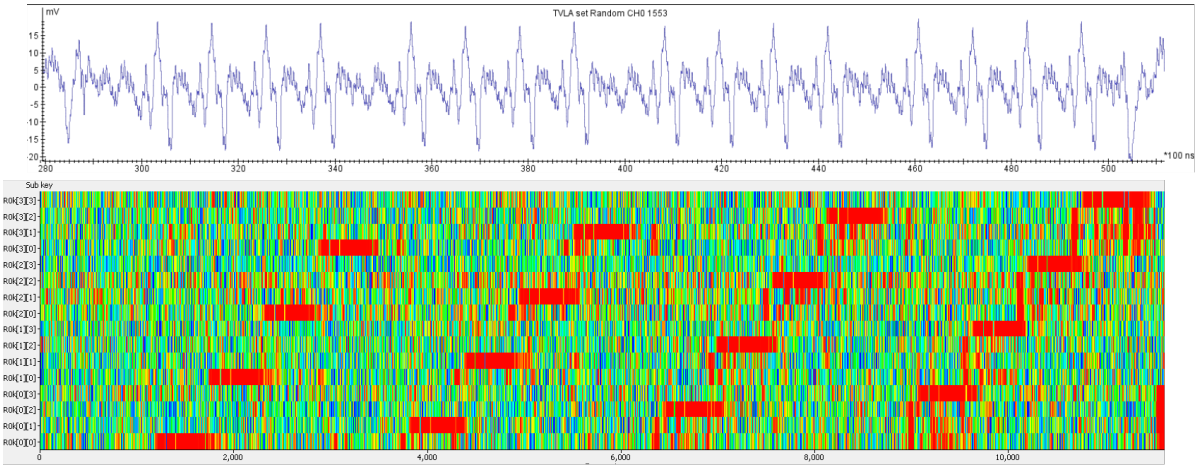


Figure 5.1: Key-ranking results on STM trace for complete s-box operation on round 1. The raw power trace showing sbox operation on 16 bytes is presented on top. The 16 rows below the trace show where the respective correct key intermediate leaks. Red shows the correct key was ranked lower (strong leaks) whereas green represents a high rank (weak leaks)

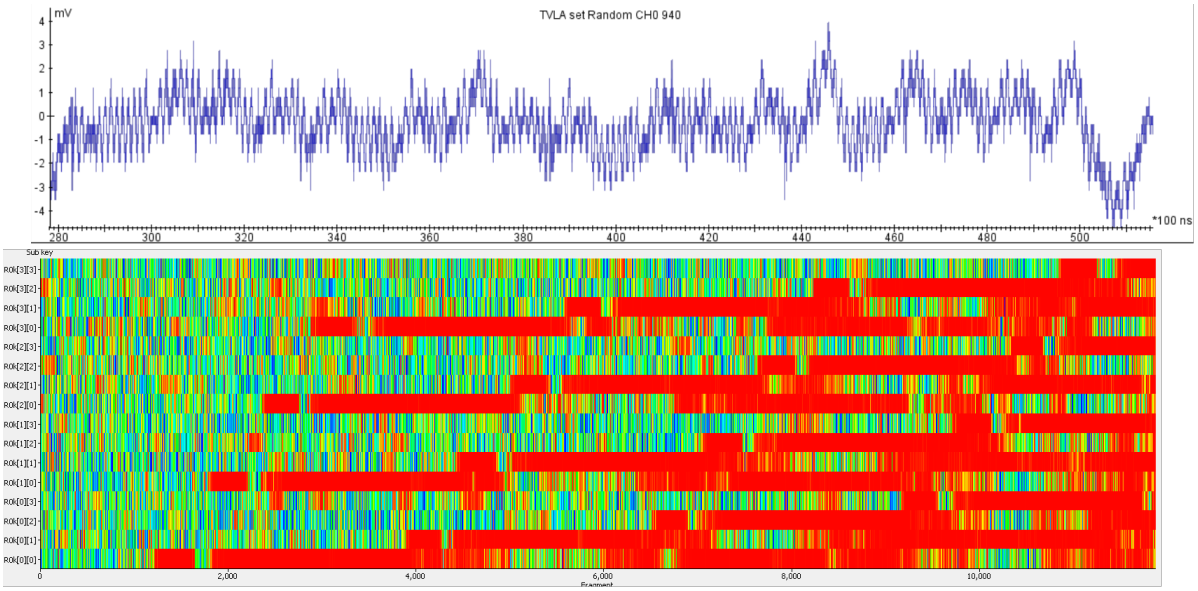


Figure 5.2: Key-ranking results on NRF trace for complete s-box operation on round 1. The raw power trace showing s-box operation on 16 bytes is presented on top. The 16 rows below the trace show where the respective correct key intermediate leaks. Red shows the correct key was ranked lower (strong leaks) whereas green represents a high rank (weak leaks)

Similarly, the results of the complete first round of S-box operation on the NRF device is presented in figure 5.2. From the NRF key-rank analysis results, we observe the correct key intermediate is leaking consistently after the first time it is read from memory. These results show that NRF device has significantly more leaks in comparison to STM class. This relates to key data leaking when operations are being performed on byte data stored in subsequent memory locations. The length of a word being parsed from memory is 4 bytes in Cortex M0. Memory access read 4 bytes of consecutive data from the provided memory address.

Comparing the leaks across the two boards' class, the leaks resulting from data-overwriting are observed at similar sections of the trace. The key bytes starts to leak subsequent to `STR` instructions, and leaks while the next byte data is loaded by `LDR` instructions.

We determine this contrasting behavior is originating from the design choices made by manufacturers. The NRF board is a low power board; memory access consumes significant power and impacts dynamic power consumption, leading to leaks observed from the NRF board. The choice of memory technology will impact the leaks observed from the board.

For the remainder of the report, we will be comparing leaks from s-box operation on 1 byte of data. The first round `S-box` operation on the first data byte is chosen to apply the above methodology. The reason for choosing `S-box` operation is due to its non-linear nature, which makes it an ideal target for SCA attacks. The selected trace section of `S-box` operation starts at index `14910` and has a length `1235` samples in the collected trace-sets. Figure 5.3 highlights the selected section of traces in raw traces. The same trace section will be used for all comparisons to maintain uniformity in the analysis.

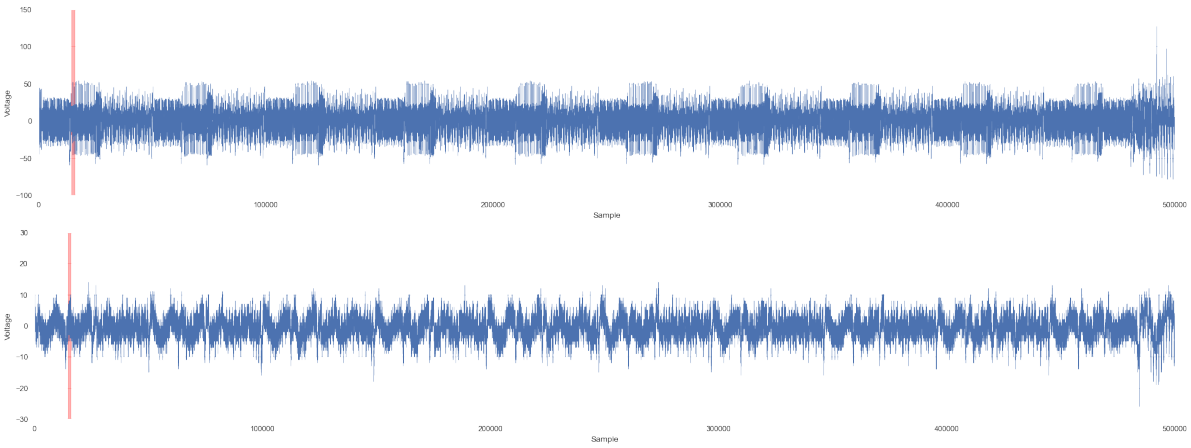


Figure 5.3: The raw power trace for STM board (above) and NRF board (below). The highlighted section marks the 1st round S-box operation on first byte of data. This section of trace has been used to profile for leaks

For a objective comparison between TVLA and KL-Divergence, we use the results of key-rank analysis. TVLA methodology has its shortcomings as described in A Cautionary Note Regarding the Usage of Leakage Detection Tests in Security Evaluation [46].

To compare trace-sets across class, board, key, and input data; the methodology is applied at different abstraction levels. Different profiling measures' reaction to variations in trace-set parameters is noted and used to comment on the similarity/dissimilarity across trace-sets.

In the next section, the methodology mentioned above is used to compare the boards from two classes of devices.

5.2. STM Intra-board comparison

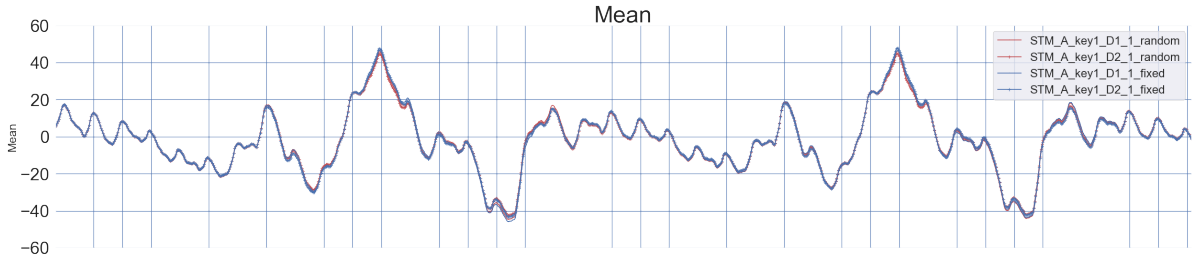
The results of applying power profile comparison on the first byte S-box operation from STM board A are presented in this section. Inter-key and inter-data comparison is presented separately, to control the influence on operand data. Key data is set while compiling code and is stored in the flash memory. This data contributes to an operand of s-box operation. The input data is provided during run-time and is present in the CPU register, forming the other operand for s-box operation.

5.2.1. STM inter-data comparison

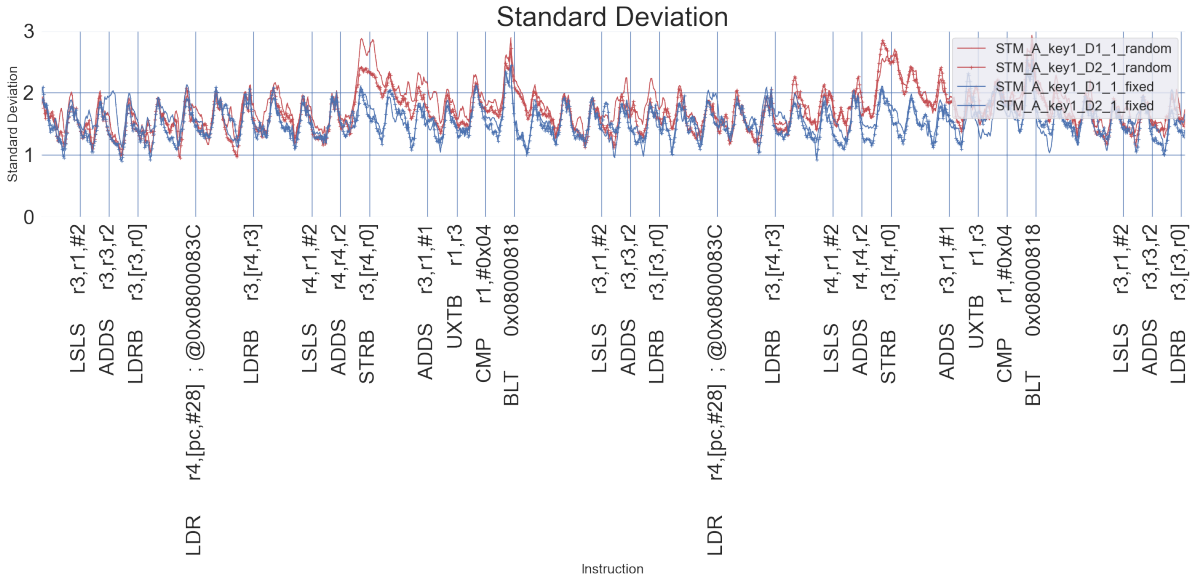
In this section, we answer how the chosen fixed set influences the measures in our methodology. For presenting the results of inter-data comparison for STM class, we will be using `STM_A_key1_D1_1/2` and `STM_A_key1_D2_1/2` trace-sets, two repetitions acquired from STM board A with the same key data but different fixed set input data.

Power profile comparison

By performing a power profile comparison, we analyze how the trace sets differ statistically. We perform a fixed vs. random comparison to check for trace sections having a dependency on input data. The fixed traces are shown in blue, while random traces are shown in red; the fixed input set is shown with a line marker. The power profile analysis results on inter-data sets from STM board A are presented in figure 5.4.



(a) The mean execution for fixed input (in blue) and random input (in red) traces of STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_B_key2_D1_1/2).



(b) The standard deviation for fixed input and random input traces of STM board A with data 1 (STM_A_key1_D1_1) and data 2 (STM_B_key2_D1_1).

Figure 5.4: Results of power profile comparison for inter-board trace-sets of STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_B_key2_D1_1/2). cycle accurate instruction labels are present in the standard deviation plot

The mean plot (figure 5.4a) shows near similar execution for trace sets; minor deviations can be seen between the fixed versus random input sets near `STRB` instruction, and subsequently at `CMP` and `BLT` instruction.

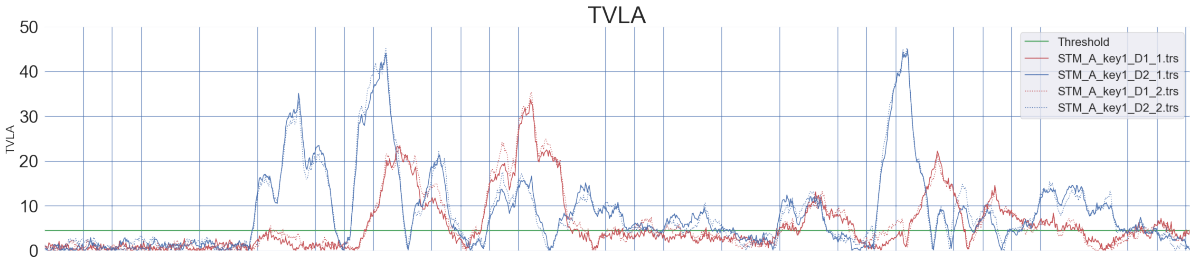
Looking at the standard deviation results in figure 5.4b we observe the standard deviation changes repetitively every clock cycle for the fixed set traces (shown in blue). Random sets (shown in red) have an identical cyclic behavior for the most part. There is a distinct increase in the standard deviation of random sets at `STRB r3,[r4,r0]` instruction, exposing a dependency on the input data. That instruction saves the value in `R3` (substituted s-box value) to memory address saved in `R4` offset by `R0` units. Subsequently, the same behavior is observed at `CMP` and `BLT` instructions also, even though these instructions control the loop iteration and do not operate on sensitive information. `LDRB [r3,r4,r3` instruction loads the substituted value to `R3` also show an increase in standard deviation for random sets, though it is not as

high. We note that the standard deviation plot for fixed input traces of the inter-data sets is similar in shape; the same behavior was observed in the STM board.

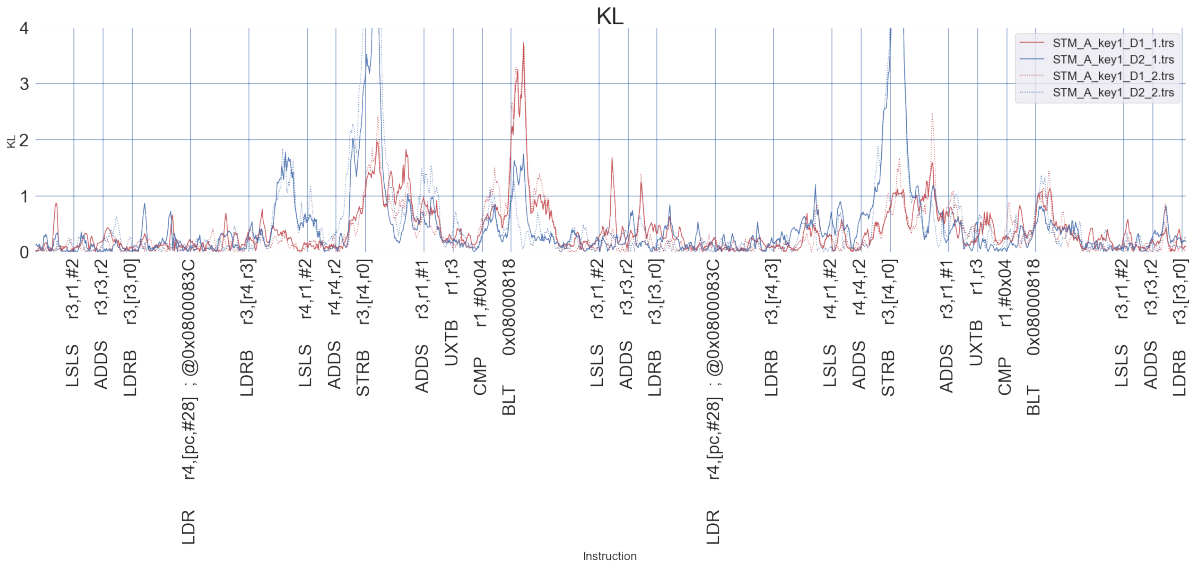
Based on the above information, some form of information leaks at the instruction highlighted by Standard deviation, though it may not necessarily be correct key data.

Leakage detection methodologies

TVLA and KL-divergence are used to check for leakages. TVLA methodology requires performing two repetitions; the repetition set results with the same fixed inputs are also presented with a dotted line. The results for data leakage comparison on inter-data trace-sets from STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_A_key1_D1_1/2) are presented in figure 5.5.

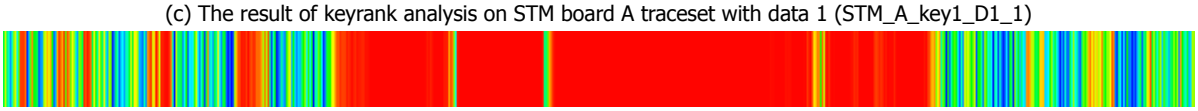


(a) The TVLA results of STM board A with fixed data 1 (STM_A_key1_D1_1/2) (shown in red) and fixed data 2 (STM_A_key1_D1_1/2) (shown in blue).



(b) The standard deviation for fixed input and random input traces of STM board A with data 1 (STM_A_key1_D1_1/2) and data 2 (STM_A_key1_D1_1/2).

(c) The result of keyrank analysis on STM board A traceset with data 1 (STM_A_key1_D1_1)



(d) The result of keyrank analysis on STM board A traceset with data 2 (STM_A_key1_D1_1)

Figure 5.5: Results of data leakage comparison for inter-board trace-sets of STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_B_key2_D1_1/2). cycle accurate instruction labels are present in the standard deviation plot

The fixed vs random (non specific) TVLA results from the two sets are shown in figure 5.5a,

It is easy to observe how the TVLA results differ based on the chosen input data; the peaks are observed at different regions for the two trace sets varying in input data. The repetition results are close to each other for the same fixed data; TVLA gives similar results for a chosen fixed set. The leaky sections shown by TVLA can vary significantly from the fixed-set choice.

KL-divergence results are presented in figure 5.5b; it can be observed that curves from the two sets are at similar locations, but they differ in amplitude, especially at `LDRB` instruction. The repetition sets also follow a similar trend. The results of KL-divergence show lesser variations compared to TVLA and have peaks at similar sections.

The results of applying key-rank analysis are shown in figure 5.5c for STM_A_key1_D1_1 and figure 5.5d for STM_A_key1_D1_1. The correct intermediate data is found to leak subsequently to `BLT` instruction, after which operations are being performed on the second key byte. This points towards the leaks being sources due to register overwrite. The key-rank analysis results show that leakage is more widespread than the regions detected by TVLA and KL divergence. Both the methodologies fail to give coverage in detecting leakage of the correct key byte at load and arithmetic instructions.

Comparing trace sets for differences

In this section, we perform specific testing on the trace sets, where fixed vs. fixed comparison (shown in blue) is performed using KL-Divergence to check for the sections where trace sets differ based on input data. The results of applying KL-divergence on random sets from both the trace sets are also presented in red; this gives a baseline idea of regions where we find differences in trace sets irrespective of input data.

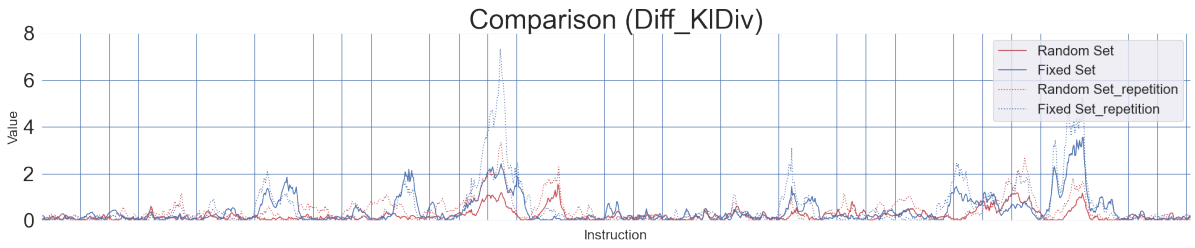


Figure 5.6: The result of specific testing on traces from STM board A with data 1 (STM_A_key1_D1_1/2) and data 2 (STM_A_key1_D1_1/2). Fixed vs fixed results shown in blue, random vs random results shown in red. Repetition sets results with dotted lines

KL divergence comparison between the trace sets is presented in figure 5.6 It shows that trace sets with a fixed input differ in probability distribution at load, store, and branch instructions. The baseline marked by random vs. random comparison shows minimal differences at load and store instructions but gives peaks near branch instructions indicating differences in pdf.

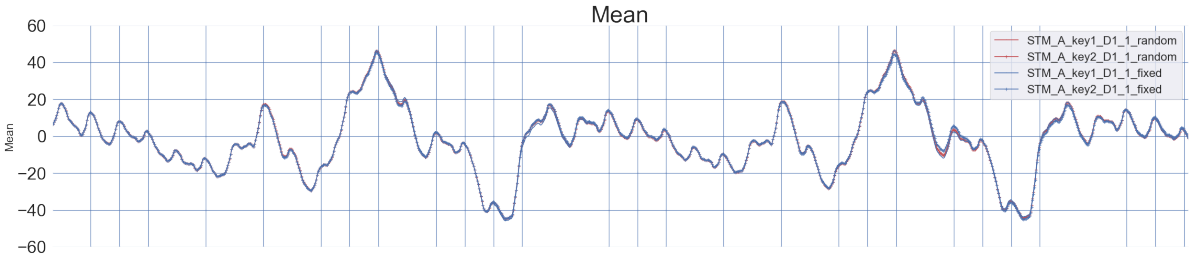
5.2.2. STM inter-key comparison

In this section, we answer how the chosen encryption key influences the trace sets for our implementation. For presenting the results of inter-key comparison for STM class, we will be using `STM_A_`**`key1`**`_D1_1/2` and `STM_A_`**`key2`**`_D1_1/2` trace-sets, two repetitions acquired from STM board A with the same key but two different input data.

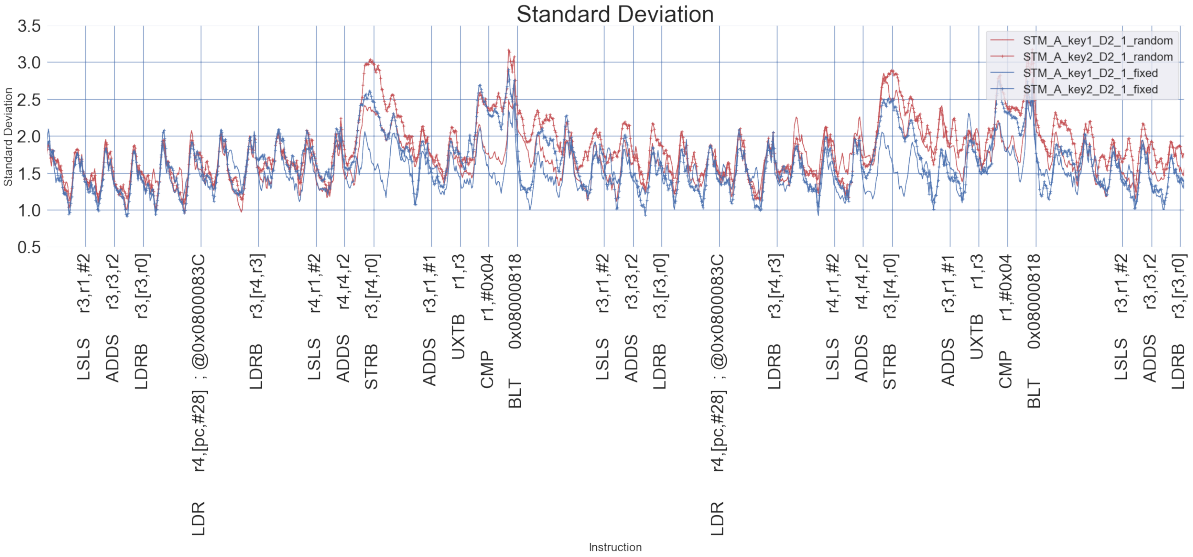
Power profile comparison

By performing a power profile comparison, we analyze how the trace sets differ statistically. We perform a fixed vs. random comparison to check for trace sections having a dependency on key data. The fixed traces are shown in blue, while random traces are shown in red; the

fixed input set is shown with a line marker. The power profile analysis results on inter-key sets from STM board A are presented in figure 5.7.



(a) The mean execution for fixed input (in blue) and random input (in red) traces of STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_B_key2_D1_1/2).



(b) The standard deviation for fixed input and random input traces of STM board A with data 1 (STM_A_key1_D1_1) and data 2 (STM_A_key2_D1_1).

Figure 5.7: Results of power profile comparison for inter-board trace-sets of STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_A_key2_D1_1/2). cycle accurate instruction labels are present in the standard deviation plot

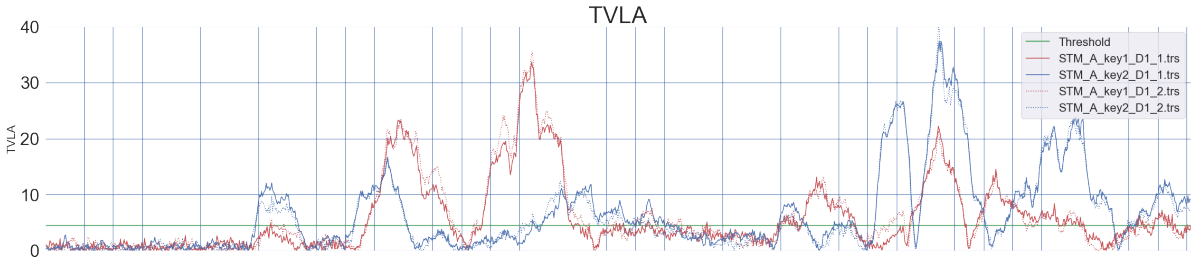
The mean plot (figure 5.7a) shows near similar execution for trace sets; minor deviations can be seen between the fixed versus random input sets near `STRB` instruction, and subsequently at `CMP` and `BLT` instruction. This is similar to the behavior observed in inter-data mean plot in figure 5.4a.

Looking at the standard deviation results in figure 5.7b we observe the same repetitive behavior every clock cycle. A line marker has been used to distinguish the plots for trace-set where key 2 was used. The red lines show the random set results, and the blue lines show the results of the fixed set. The standard deviation in random set increases with respect to coresponding fixed sets at `STRB`, `CMP`, `BLT` and `LDR` instructions. This behavior is the same as observed in the inter-data standard deviation trace in figure 5.4b. An interesting observation to make is that the choice of key influences the standard deviation. In the inter-data comparison results where the key was same, we observe both the fixed sets had identical behaviors. The standard deviation plot for fixed input traces of the inter-key sets differ in shape; In inter-key standard deviation results for trace set with key 2 (STM_A_key2_D1_1) are significantly higher than what was observed for key 1 (STM_A_key1_D1_1). This behavior is observed throughout sections where random sets' standard deviation goes higher .

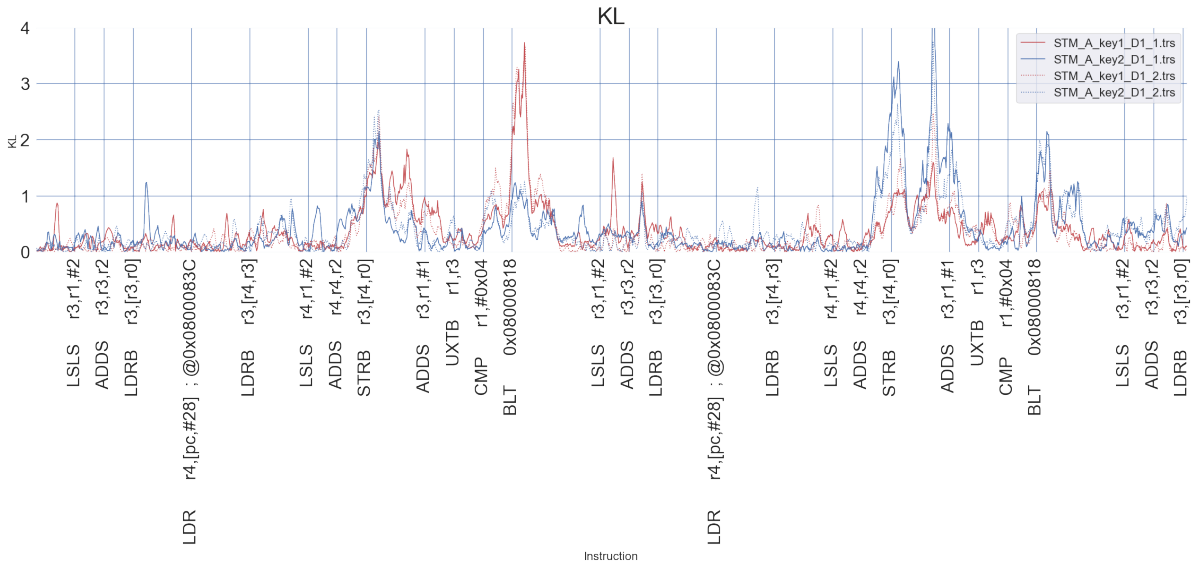
Based on the above, information relating to key byte does leak at the instruction highlighted by the difference in Standard deviation.

Leakage detection methodologies

TVLA and KL-divergence are used to check for leakages. The results of repetition sets have also been included, shown with a dotted line. The results for data leakage comparison on inter-key trace-sets from STM board A with key 1 (STM_A_key1_D1_1/2) (red) and key 2 (STM_A_key2_D1_1/2) (blue), keeping the fixed input data same are presented in figure 5.8.



(a) The TVLA results of STM board A with fixed data 1 (STM_A_key1_D1_1/2) (shown in red) and fixed data 2 (STM_A_key1_D1_1/2) (shown in blue).



(b) The standard deviation for fixed input and random input traces of STM board A with data 1 (STM_A_key1_D1_1/2) and data 2 (STM_A_key1_D1_1/2).

(c) Results for leak of first byte data using keyrank analysis on STM board A traceset with key 1 (STM_A_key1_D1_1)

(d) Results for leak of first byte data using keyrank analysis on STM board A traceset with key 2 (STM_A_key2_D1_1)

Figure 5.8: Results of data leakage comparison for inter-board trace-sets of STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_B_key2_D1_1/2). cycle accurate instruction labels are present in the standard deviation plot

The fixed vs random (non specific) TVLA results from the two sets are shown in figure 5.8a, It is easy to observe how the TVLA results are grouped based on the chosen key; the peaks differ in regions and vary in amplitudes. The repetition results are close to each other for the same keys; TVLA results show different leakage intensities based on key data.

KL-divergence results are presented in figure 5.8b; peaks from the two sets are at similar locations, but they differ in amplitude. The repetition sets also follow a similar trend. The results of KL-divergence show lesser variations compared to TVLA and have peaks at similar sections. This implies that measuring the probability distributions' difference can be a more uniform measure to check for leakages.

The results of applying key-rank analysis are shown in figure 5.8c for STM_A_key1_D1_1 and figure 5.8d for STM_A_key2_D1_1. Both the keys leak at a similar section of power trace, there are minor variations in the intensity of leaks. These results were calculated on a trace set with 1.2k random input traces for each key.

Comparing trace sets for differences

This section performs specific testing on the inter-key traces, where fixed vs. fixed comparison (shown in blue) is performed using KL-Divergence to find sections where trace sets differ based on key data. The results of applying KL-divergence on random sets inter-key traces is also presented in red; this gives a baseline idea of regions where we find differences in trace sets irrespective of input data.

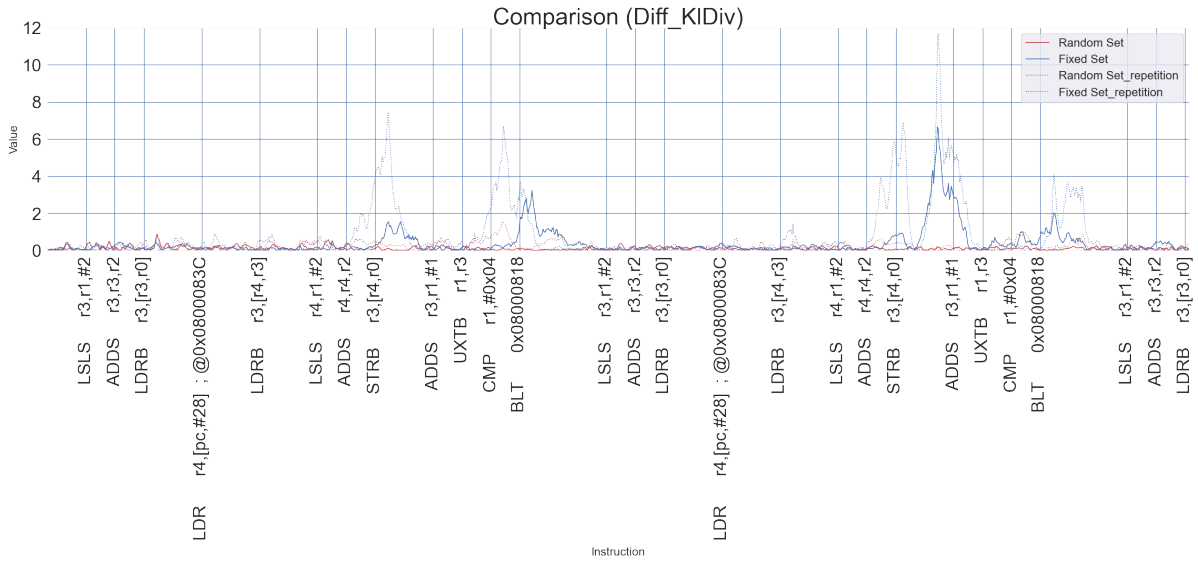


Figure 5.9: The result of specific testing on traces from STM board A key 1 (STM_A_key1_D1_1/2) and key 2 (STM_A_key2_D1_1/2). Fixed vs fixed results shown in blue, random vs random results shown in red. Repetition sets results with dotted lines

KL divergence comparison between the trace sets is presented in figure 5.9 It shows that trace sets with a fixed input differ in probability distribution at store and branch instructions. This differs from the results of inter-data comparison in figure 5.6, where `LDRB` instruction also differed. The baseline marked by random vs. random comparison shows minimal differences at load and store instructions but gives small peaks at `STRB` and `CMP`.

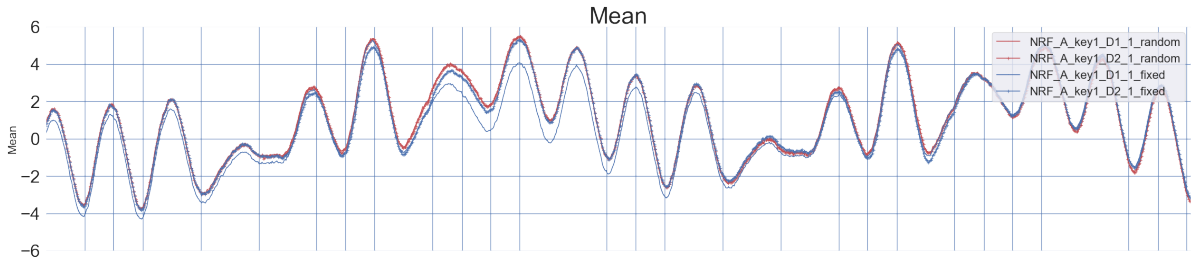
5.3. NRF intra-board comparison

5.3.1. NRF inter-data comparison

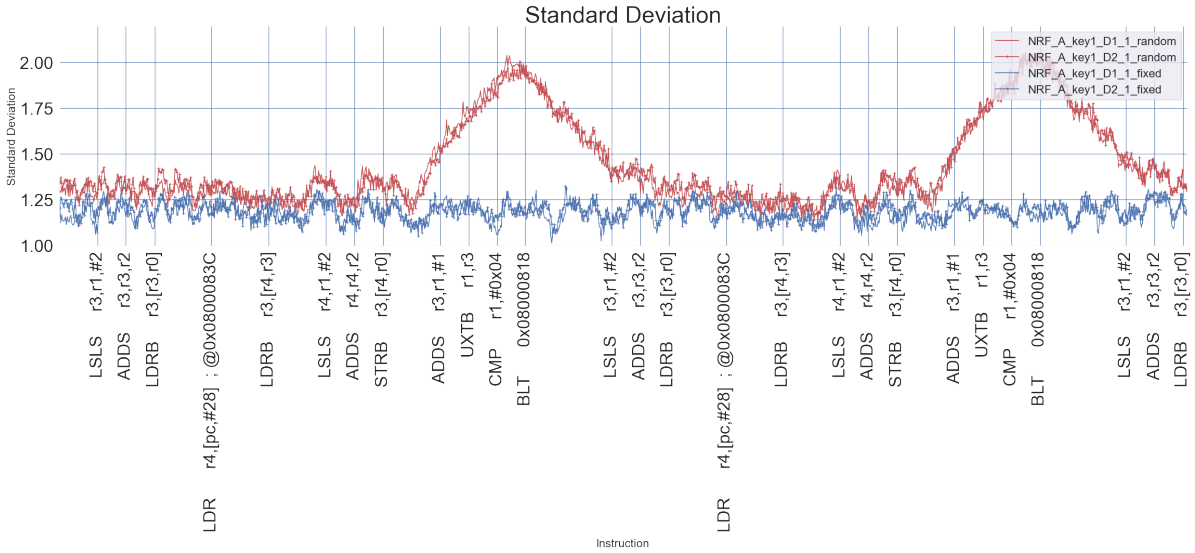
To present the results of the NRF inter-key comparison, we will be using `NRF_A_key1_`**`D1`**`_1/2` and `NRF_A_key1_`**`D2`**`_1/2` trace-sets, acquired from NRF board A with the same key data but different fixed set data.

Power profile comparison

By performing a power profile comparison, we analyze how the trace sets differ statistically. We perform a fixed vs. random comparison to check for trace sections having a dependency on input data. The fixed traces are shown in blue, while random traces are shown in red; the fixed input set is shown with a line marker. The power profile analysis results on inter-data sets from STM board A are presented in figure 5.10.



(a) The mean execution for fixed input (in blue) and random input (in red) traces of NRF board A with fixed data 1 (NRF_A_key1_D1_1/2) and fixed data 2 (NRF_A_key2_D1_1/2).



(b) The standard deviation for fixed input and random input traces of NRF board A with data 1 (NRF_A_key1_D1_1) and data 2 (NRF_A_key2_D1_1).

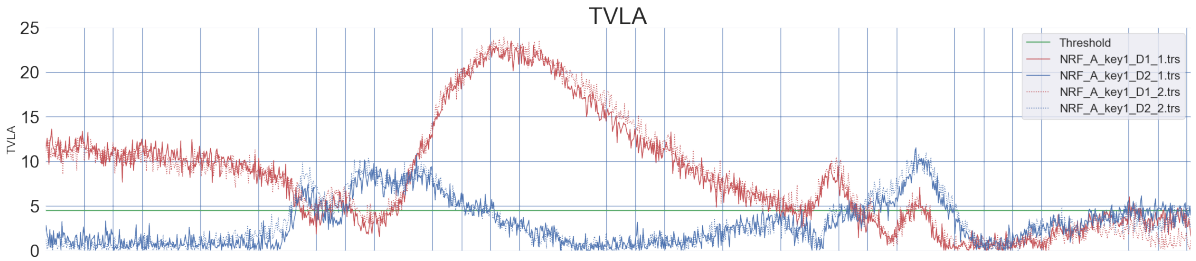
Figure 5.10: Results of power profile comparison for inter-data trace-sets of NRF board A with fixed data 1 (NRF_A_key1_D1_1/2) and fixed data 2 (NRF_A_key1_D2_1/2). cycle-accurate instruction labels are present in the standard deviation plot

The mean plot for fixed and random components of inter-data NRF traces is shown in figure 5.10a. It can be observed that the random input sets have a near similar execution trace. However, there are deviations between the mean traces for fixed input sets, after `STRB r3,[r4,r0]` instruction lasting until the next `LDRB` instruction.

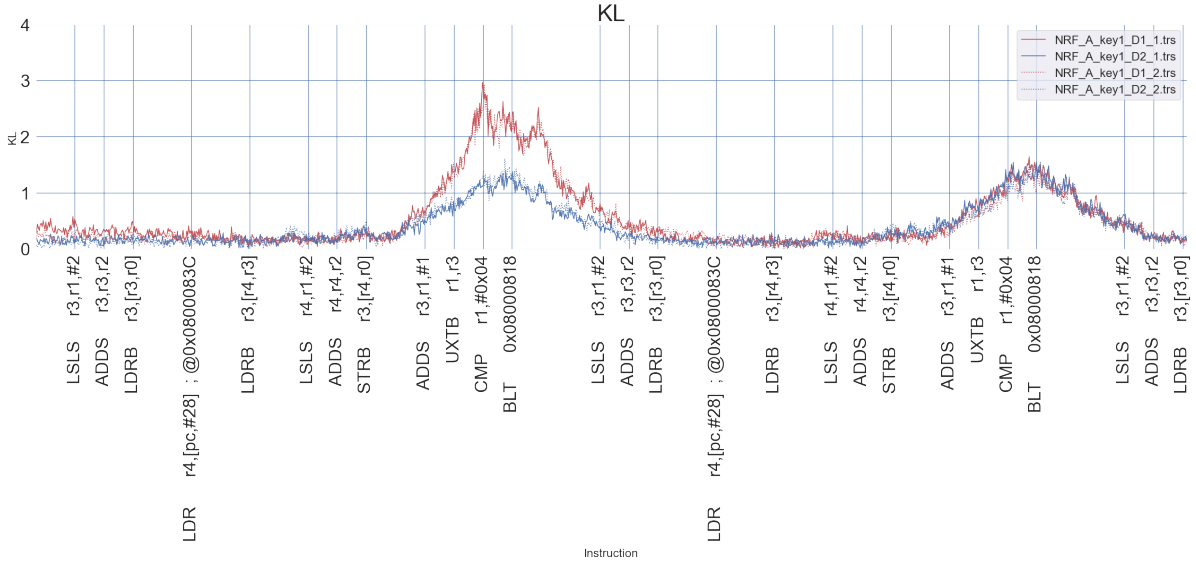
For the same section of trace, the results of standard deviation in figure 5.10b, the standard deviation of random set showed an increase compared to the standard deviation of fixed sets. For NRF board, the standard deviation of random sets is higher than the standard deviation of fixed set throughout the trace, which is evidence of leaks. It can be concluded intensity of leaked information increases significantly subsequent to `STRB r3,[r4,r0]` instruction. We note that the standard deviation plot for fixed input traces of the inter-data sets is similar in shape; the same behavior was observed in the STM board.

Leakage detection methodologies

TVLA and KL-divergence are used to check for leakages. TVLA methodology requires performing two repetitions; the repetition set results with the same fixed inputs are also presented with a dotted line. The results for data leakage comparison on inter-data trace-sets from NRF board A with fixed data 1 (NRF_A_key1_D1_1/2) and fixed data 2 (NRF_A_key1_D1_1/2) are presented in figure 5.11.



(a) The standard deviation plot for random traces from NRF board A fixed set 1 (NRF_A_key1_D1_1/2) and fixed set 2 (NRF_A_key1_D2_1/2).



(b) The standard deviation plot for random traces from NRF board A fixed set 1 (NRF_A_key1_D1_1/2) and fixed set 2 (NRF_A_key1_D2_1/2).

(c) Key-ranking results from NRF board A fixed set 1 (NRF_A_key1_D1_1).

(d) Key-ranking results from NRF board A fixed set 2 (NRF_A_key1_D1_1).

Figure 5.11: Results of data leakage comparison for inter-board trace-sets from NRF board A fixed set 1 (NRF_A_key1_D1_1/2) and fixed set 2 (NRF_A_key1_D2_1/2).

The TVLA results from the two sets are shown in figure 5.11a; we find the two fixed sets lead to entirely different TVLA results; this is similar to what was observed in STM boards. The repetition results are close to each other for the same fixed data, though the shape of curves varies significantly from the fixed input set choice. The choice of input data influences the intensity of leaks.

KL-divergence results are presented in figure 5.11b; The results for both the set are at the same location but with varying intensities. The results of repetitions are indistinguishable. The shape of curve is different from that obtained from TVLA.

The key-ranking results on fixed data set 1 in figure 5.11c and fixed data 2 in figure 5.11d,

we find a similar behavior for leakage of correct key byte for both sets.

We find the data leakage predictions from TVLA on (NRF_A_key1_D2_1/2) different from the key-ranking results. The results from KL-divergence do present peaks at locations coinciding with key-ranking results.

Comparing trace sets for differences

In this section, we perform specific testing on the trace sets, where fixed vs. fixed comparison (shown in blue) is performed using KL-Divergence to check for the sections where trace sets differ based on input data. The results of applying KL-divergence on random sets from both the trace sets are also presented in red; this gives a baseline idea of regions where we find differences in trace sets irrespective of input data.

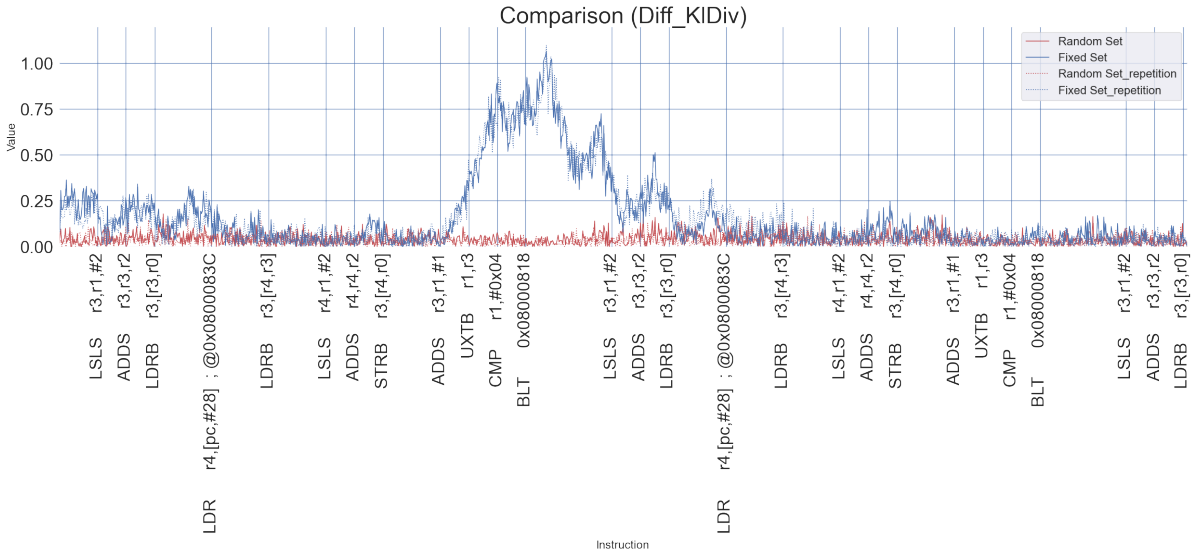


Figure 5.12: The result of specific testing on traces from NRF board A with data 1 (NRF_A_key1_D1_1/2) and data 2 (NRF_A_key1_D2_1/2). Fixed vs fixed results shown in blue, random vs random results shown in red. Repetition sets results with dotted lines

KL divergence comparison between the trace sets is presented in figure 6.6 It shows that trace sets with a fixed input differ in probability distribution after `STRB` instruction until `LDRB` instruction. This region coincides with the leaky section.

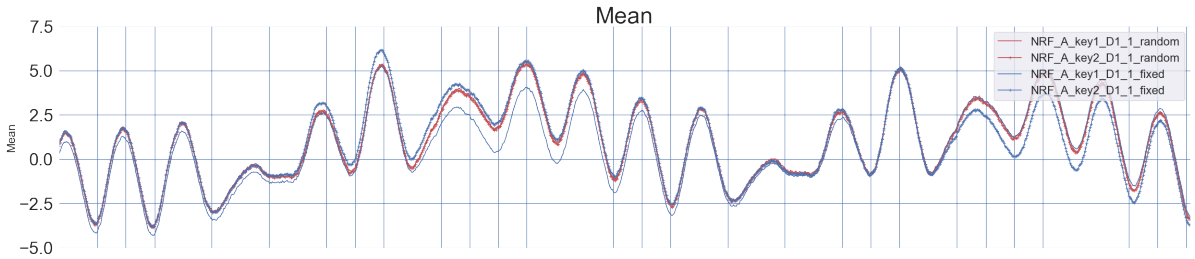
5.3.2. NRF inter-key comparison

For presenting the results of inter-key comparison for NRF class, we will be using `NRF_A_`**`key1`**`_D1_1/2` and `NRF_A_`**`key2`**`_D1_1/2` trace-sets, acquired from NRF board A with different key data but same fixed set data. The results of the power profile comparison are presented in figure 5.13.

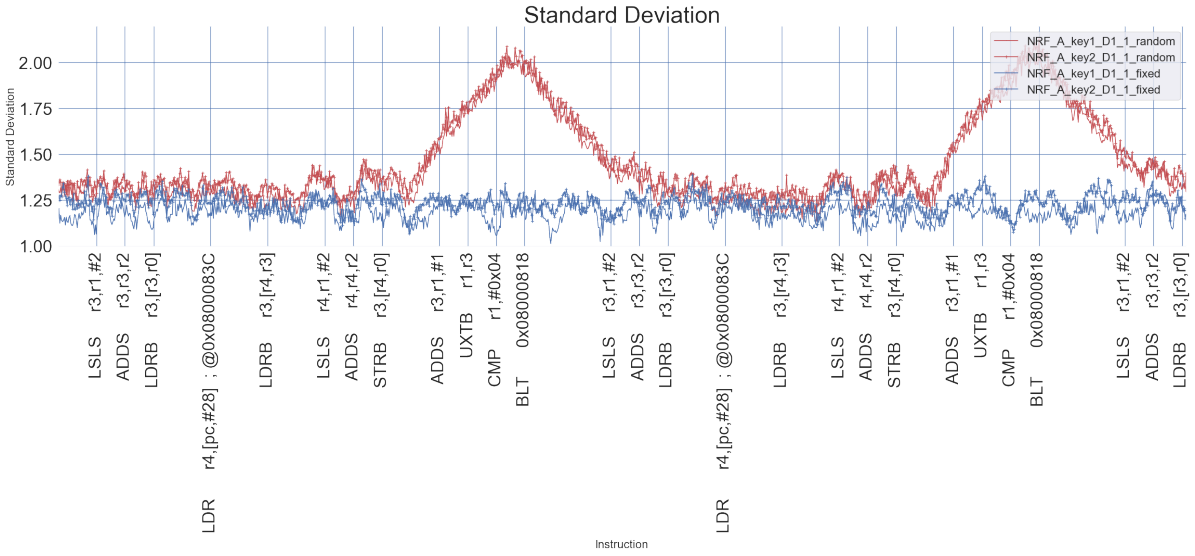
Power profile comparison

The mean plot for fixed and random component of inter-data NRF traces is shown in figure 5.10a. It can be observed that the random input sets have a near similar execution trace. However, there are deviations between the mean traces for fixed input sets, after `STRB r3,[r4,r0]` instruction lasting until the next `LDRB` instruction. This section is similar to what was observed in inter-data effects, unlike the STM board, where we observe different results. Due to the low SNR of NRF trace, the observed leakage is visible as one diffused peak after `STRB` instruction.

The standard deviation plots for fixed vs random input inter-key sets are presented in figures 5.13b. We note that the standard deviation plot for fixed input traces of the inter-key



(a) The mean execution for fixed input (in blue) and random input (in red) traces of NRF board A with fixed data 1 (NRF_A_key1_D1_1/2) and fixed data 2 (NRF_A_key2_D1_1/2).



(b) The standard deviation for fixed input and random input traces of NRF board A with data 1 (NRF_A_key1_D1_1) and data 2 (NRF_A_key2_D1_1).

Figure 5.13: Results of power profile comparison for inter-board trace-sets of NRF board A with fixed data 1 (NRF_A_key1_D1_1/2) and fixed data 2 (NRF_A_key2_D1_1/2). cycle accurate instruction labels are present in the standard deviation plot

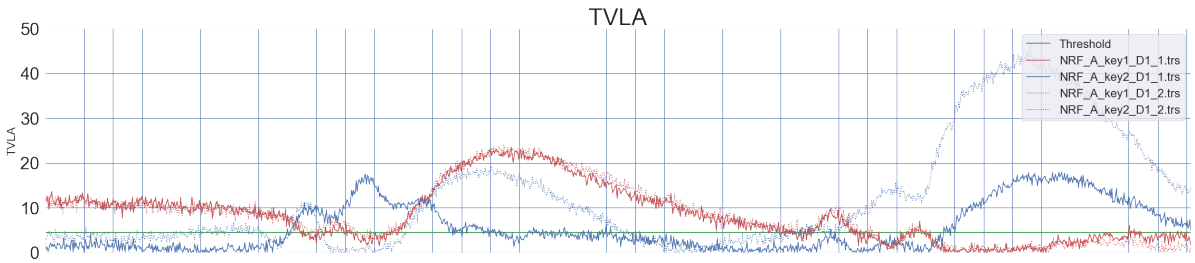
sets differs in shape; the same behavior was observed in the STM board. We observe an increase in the standard deviation of random sets at the instructions where data dependency exists, though this does relate to the data dependency of operations. The key-data effects are observed in standard deviation traces and are not influenced by fixed input set used for comparison.

Leakage detection methodologies

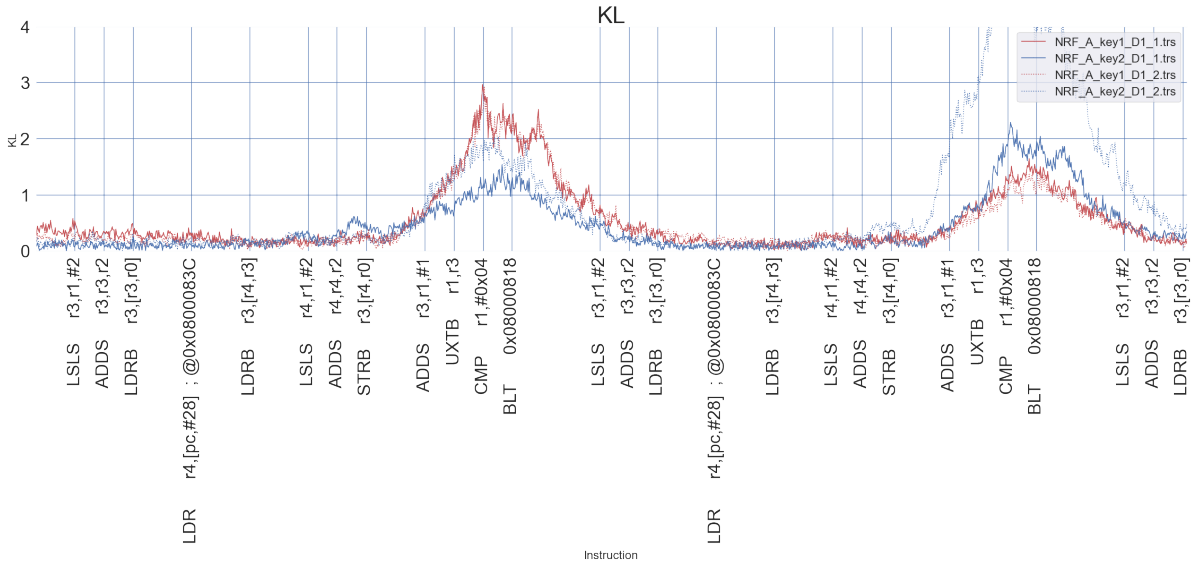
The results of data leakage comparison for inter-key trace-sets from NRF board A with key 1 (NRF_A_key1_D1_1/2) and key 2 (NRF_A_key2_D1_1/2) are presented in figure 5.14.

The TVLA results from the inter-key sets are shown in 5.14a. The repetition results on TVLA for key 2 give different results due to inconsistency in the data set. The TVLA results show that the for two keys differ significantly.

From the results of KL-divergence shown in figure 5.14b, we observe peaks at the same locations from all the trace-set though of varying amplitudes. Comparing the results of KL-divergence and key-rank analysis, the curves are at the regions where correct key-data is leaking. The leaks section observed from key-ranking results is wider for key 1 shown in figure 5.14c compared to key 2 shown in figure 5.14d; this relates to the amplitude peaks from KL-divergence results.



(a) The TVLA results of NRF board A with fixed data 1 (NRF_A_key1_D1_1/2) (shown in red) and fixed data 2 (NRF_A_key1_D1_1/2) (shown in blue).



(b) The standard deviation for fixed input and random input traces of NRF board A with data 1 (NRF_A_key1_D1_1/2) and data 2 (NRF_A_key1_D1_1/2).

(c) Results for leak of first byte data using keyrank analysis on NRF board A traceset with key 1 (NRF_A_key1_D1_1)

(d) Results for leak of first byte data using keyrank analysis on NRF board A traceset with key 2 (NRF_A_key2_D1_1)

Figure 5.14: Results of data leakage comparison for inter-board trace-sets of NRF board A with fixed data 1 (NRF_A_key1_D1_1/2) and fixed data 2 (NRF_B_key2_D1_1/2). cycle accurate instruction labels are present in the standard deviation plot

Comparing trace sets for differences

In this section, we perform specific testing on the inter-key traces, where fixed vs. fixed comparison (shown in blue) is performed using KL-Divergence to find sections where trace sets differ based on key data. The results of applying KL-divergence on random sets inter-key traces is also presented in red; this gives a baseline idea of regions where we find differences in trace sets irrespective of input data.

KL divergence comparison between the trace sets is presented in figure 5.15 It shows that trace sets with a fixed input differ in probability distribution after store instruction until branch instructions. This result is similar to inter-data comparison in figure 5.15. Unlike STM board we do not see any interesting observations from NRF board using KL-divergence. Though KL-Divergence does works better in identifying leaky sections.

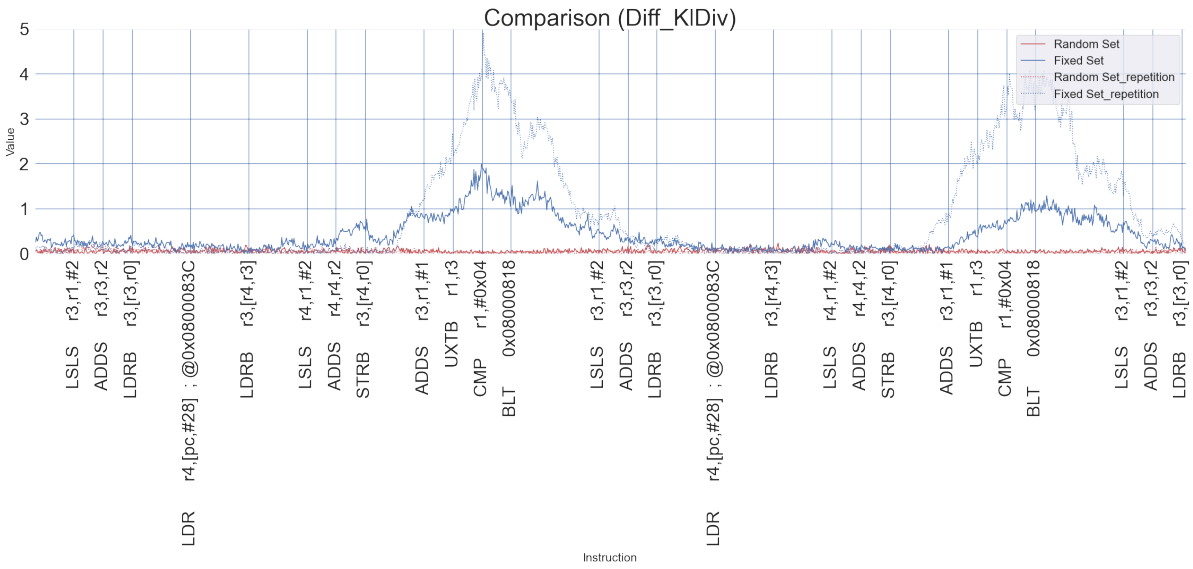


Figure 5.15: The result of specific testing on traces from NRF board A with data 1 (NRF_A_key1_D1_1/2) and data 2 (NRF_A_key2_D1_1/2). Fixed vs fixed results shown in blue, random vs random results shown in red. Repetition sets results with dotted lines

5.4. Summarising chapter

To conclude this chapter, we make the following observations

1. The leaks observed from non-specific tests using TVLA are influenced by the input data (STM figure 5.5a, NRF figure 5.11a) and the key data (STM figure 5.8a, NRF figure 5.14a) across boards of both class. This behaviour was observed

2. Key data has an influence on the standard deviation in our implementation, standard deviation was not affected by the choice of input fixed set. This can be observed comparing the results of standard deviation for STM inter-data (figure 5.4b) and inter-key sets (figure 5.7b); as well as in NRF inter-data (figure 5.10b) and inter-key sets (figure 5.13b.

3. The leaks detected using KL-Divergence have more uniform peaks compared to TVLA, this was observed in data leakage comparison for all test cases. KL-Divergence results show consistency and location of curves is not influenced by the choice of input and key data, unlike TVLA. This can be observed from the results of data leakage comparison for all sets.

4. From KL-divergence on fixed input inter-data sets on STM board (in section 6.2.3), has peaks on `LDRB`, `STRB`, `CMP` and `BLT` instructions. Which correlates to the instructions operating on input byte data for our S-box implementation KL-divergence on fixed input inter-key sets shows peaks at only `STRB`, `CMP` and `BLT`, the peak at `LDRB` instruction loading input data from memory are are absent. We note that the differences of inter-data sets are more prominent than inter-key sets for STM class, we do not observe any differences in the results from NRF class. Using KL-Divergence for specific test (fixed vs fixed) can be used as a tool to compare the level of differences between two power traces.

6

Inter-board comparison

It this chapter we answer the research question how significant are the inter device variations? The inter-board trace sets have the same input and key data, but are acquired from different physical devices. To answer the research question, traces acquired from two board iterations are compared for both device classes (STM 6.1, NRF 6.2). The methodology used for comparison is similar to that in intra-device comparison (Chapter 5); in addition to power profile and data leakage comparison, templates are profiled to comment on the portability of attacks 6.3. These analyses are used to answer the research question of the similarity/dissimilarity between two boards of same device.

Manufactured silicon chips have variations due to the raw material used or variations in manufacturing process. Each silicon chip has its own power signature originating from variations at the micro-architecture level. Non-uniform etching can introduce inconsistencies in transistors' depletion layer, which will affect the leakage current generated on switching. Inconsistencies are spread-out across peripherals at the micro-architecture level; each physical device will have its power fingerprint resulting from the accumulation of these effects.

VLSI test design is the field that determines the quality of manufactured chips by measuring leakage current on the device under test to check its fitness. Using our methodology, we expect to see effects that can be used to distinguish the execution across two physical boards.

Traces from targets belonging to the same class `STM` or `NRF` but with different board iterations `Board A` and `Board B` are compared. Uniformity is maintained across the two sets by having the same key and fixed-input data. A repetition dataset is also used in order to report observations with confidence. Repetition data sets are collected, keeping all the parameters consistent, to highlight the variations that can be expected between two acquisitions.

Inter-board comparison is performed on trace-sets `X_A_keyY_DZ_1/2` and `X_B_keyY_DZ_1/2` where value of parameter in green is the board iteration which will be different whereas the value of parameters in orange is same for both trace-sets.

1. `X` can be `STM` or `NRF` representing the class of devices from Cortex-M0 family we are comparing, has to be same for devices to be from same class;

2. `Y` can be `1` or `2` representing the encryption key, needs to be same across both sets in order to have same key data;

3. `Z` can be `1` or `2` representing the fixed set, needs to be same across both sets in order to have same input data.

The results for both the device's classes are presented separately to look for similar effects at the board level for devices of both classes. We will present the results on STM inter-board comparison followed by NRF inter-board comparison.

6.1. STM inter-board comparison

In this section, we answer how the physical board iteration influences the measures in our methodology, S-box traces from two STM board iterations marked as STM-A and STM-B are used for comparing. For presenting the results of STM inter-board comparison we will be using `STM_A_key1_D1_1/2` and `STM_B_key1_D2_1/2` (shown with marker) trace-sets.

6.1.1. Power profile comparison

The results of the power profile comparison are presented in figure 6.1. The fixed-set (in blue) and random-set (in red) results are plotted for inter-board comparison, the inter-board sets can be distinguished by the marker. Comparison is being made between fixed-sets and random-sets from different physical boards to compare execution differences between boards.

The mean-plot of random sets and fixed-sets in 6.1a shows an identical shape, but we can observe deviations in amplitude relating to the board iteration. To magnify the effect of distance in means, the results for difference of means (Set B-Set A) have presented in figure 6.1b. The sections in difference of mean plots which lie above 0 imply higher power consumption by board B, while the sections below 0 imply higher power consumption by board A.

Looking at distance of mean plot, the relation of power consumption remains consistent across fixed-set and random-set, this shows that input data plays no role in the observed differences. The distance between the means of two sets oscillates every clock cycle, this implies a small drift is present in the clock of devices. The shape of difference of mean plot is identical on the subsequent instructions also, so it can be determined that same behaviour is observed for repetition of instructions.

This effect is observed for both random set and fixed set implying no dependency on input data, and can be contributed to the nature of physical devices. It can be used to measure the instructions that differ the most, quantify how the signatures vary over different instructions.

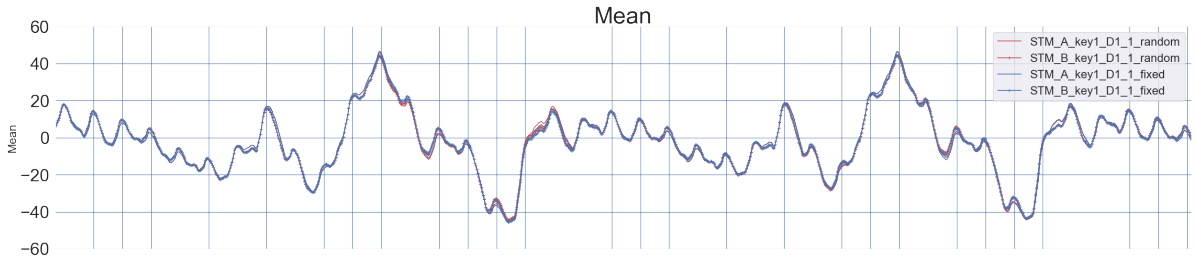
The standard deviation-plots for random-set and fixed-set are presented in figures 6.1c. From the standard deviation results, we were not able to identify effects that can help distinguish between the two boards. The standard deviation from two subsequent repetitions can have enough variations to diminish the effects of inter-board execution. We do not observe effects from standard deviation that can be used to distinguish the two physical boards.

6.1.2. Data leakage comparison

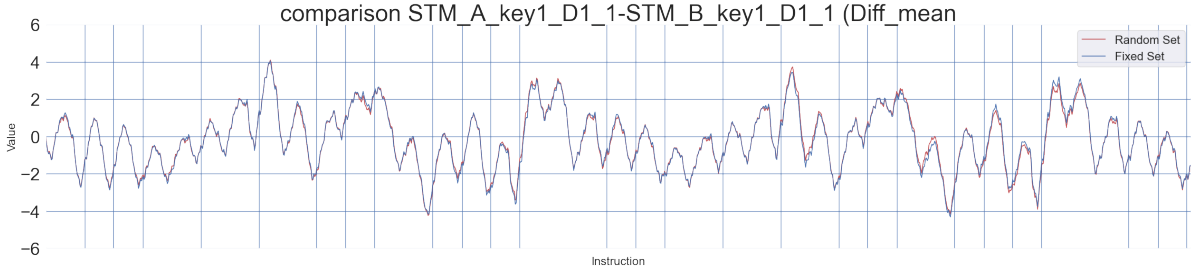
TVLA, KL-Divergence and key-rank analysis are performed to comment on data leakage from inter-board sets. TVLA methodology requires performing two repetitions; the repetition set results with the same fixed inputs are also presented with a dotted line. The results for data leakage comparison on inter-data trace-sets from STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_A_key1_D1_1/2) are presented in figure 6.2.

The TVLA results for inter-board sets are presented in 6.2a. The TVLA plot has a similar shape for acquisition on both the boards, the repetitions shown with dotted line also give the same results. This shows that the location and intensity of leaks across both boards is same.

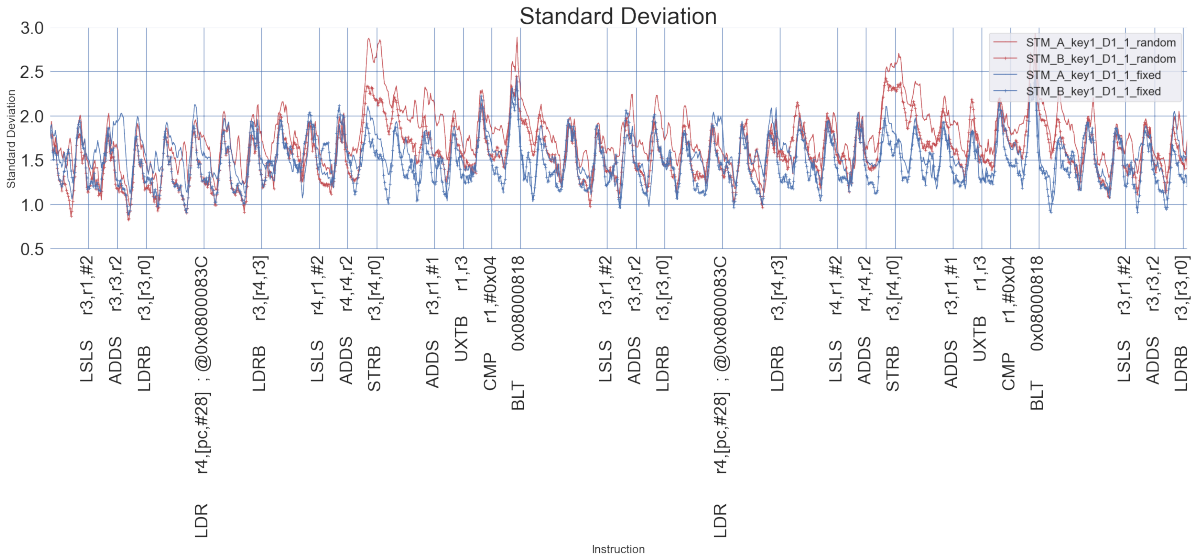
The results of KL-divergence in 6.2b shows narrower peaks corresponding to TVLA results at a similar location. Small peaks distributed along the trace are also shown in KL-divergence results, but these peaks vary across the repetition sets. The two leakage detection methodologies give us similar results across the two physical boards.



(a) The mean execution for fixed input (in blue) and random input (in red) traces of STM board A (STM_A_key1_D1_1/2) and board B (STM_B_key2_D1_1/2).



(b) The difference of mean for fixed input (in blue) and random input (in red) traces of STM board A (STM_A_key1_D1_1/2) and board B (STM_B_key2_D1_1/2).

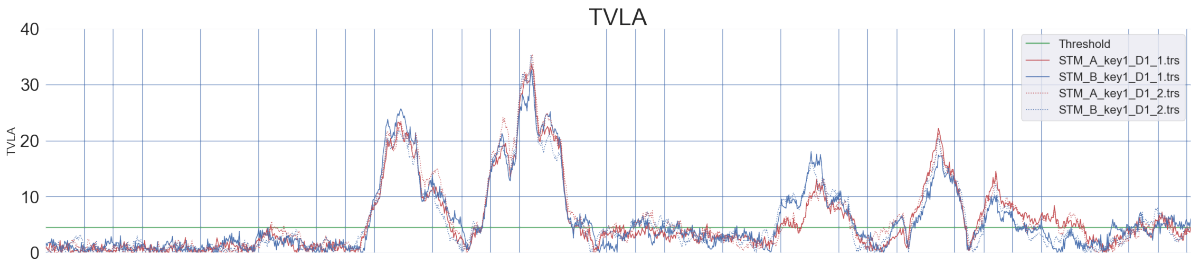


(c) The standard deviation for fixed input and random input traces of STM board A (STM_A_key1_D1_1/2) and board B (STM_B_key2_D1_1/2).

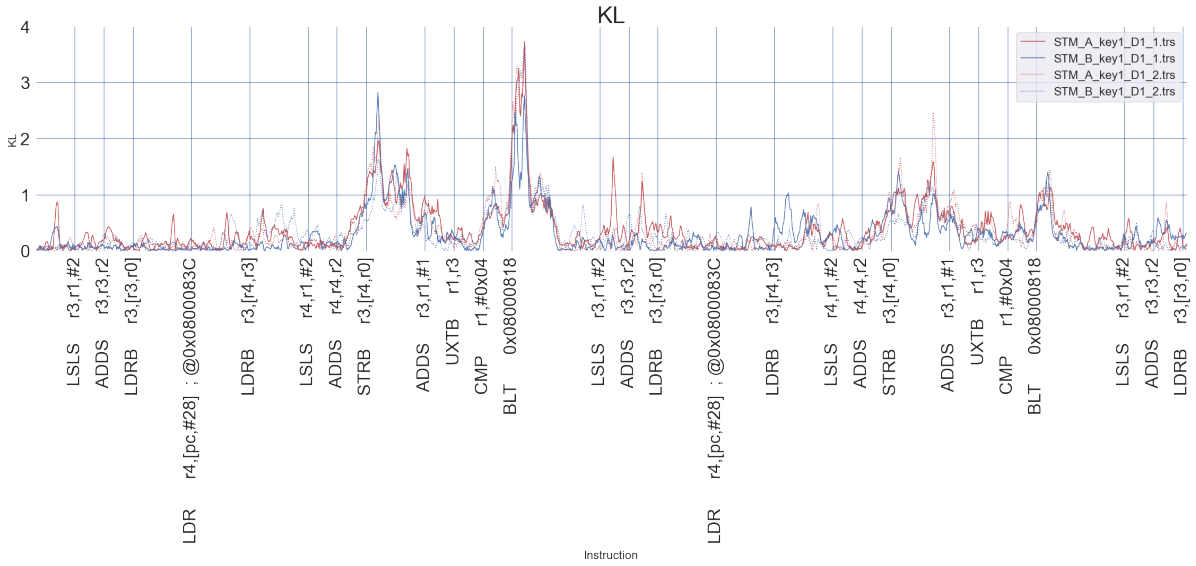
Figure 6.1: Results of power profile comparison for inter-board trace-sets of STM board A (STM_A_key1_D1_1/2) and board B (STM_B_key1_D1_1/2). cycle accurate instruction labels are present in the standard deviation plot

The results of key-rank analysis on the two boards are presented in figure 6.2c and 6.2d. Key-ranking results show leaks over a wider section of power traces, for both the boards than predicted by results of leakage detection methodologies. Key-rank results from board A have traces of leaks at the beginning of trace also, this behaviour was not seen from STM board B. These leaks can be originating due to manufacturing defects in board A that are not significant for board B. Gaps in leak are observed in both boards at same `UXTB` and `BLT` instructions, which can be sourced to effects in physical layer. Comparing the key-rank analysis results across two boards, we observe do not observe differences in leakage of correct key data.

We observe gaps in the result of key-rank analysis at `ADDS` and `BLT` instruction, which is consistent across both the boards. We can contribute this effect to operations being imple-



(a) TVLA calculated on fixed vs random traces of STM board A (STM_A_key1_D1_1/2) (shown in red) and STM board B (STM_A_key1_D1_1/2) (shown in blue).



(b) KL-Divergence calculated on fixed vs random traces of STM board A (STM_A_key1_D1_1/2) (shown in red) and STM board B (STM_A_key1_D1_1/2) (shown in blue).

(c) The result of keyrank analysis on STM board A traceset with data 1 (STM_A_key1_D1_1)

(d) The result of keyrank analysis on STM board B traceset with data 1 (STM_B_key1_D1_1)

Figure 6.2: Results of data leakage comparison for inter-board trace-sets of STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_B_key2_D1_1/2). cycle accurate instruction labels are present in the standard deviation plot

mented at the hardware level, which mask leakage of key data at those locations.

6.1.3. Comparing trace sets for differences

In this section, we perform specific testing on the trace sets, where fixed vs. fixed comparison (shown in blue) is performed using KL-Divergence to check for the sections where trace sets differ based on input data. The results of applying KL-divergence on random sets from both the trace sets are also presented in red; this gives a baseline idea of regions where we find differences in trace sets irrespective of input data.

KL divergence comparison between the trace sets is presented in figure 6.3 It shows that trace sets acquired from 2 boards differ significantly comparing it to other KL-divergence comparisons for STM inter-data (figure 5.6) and inter-key (figure 5.9). These results show that inter-board traces have a difference in probability distribution every clock cycle. Comparing

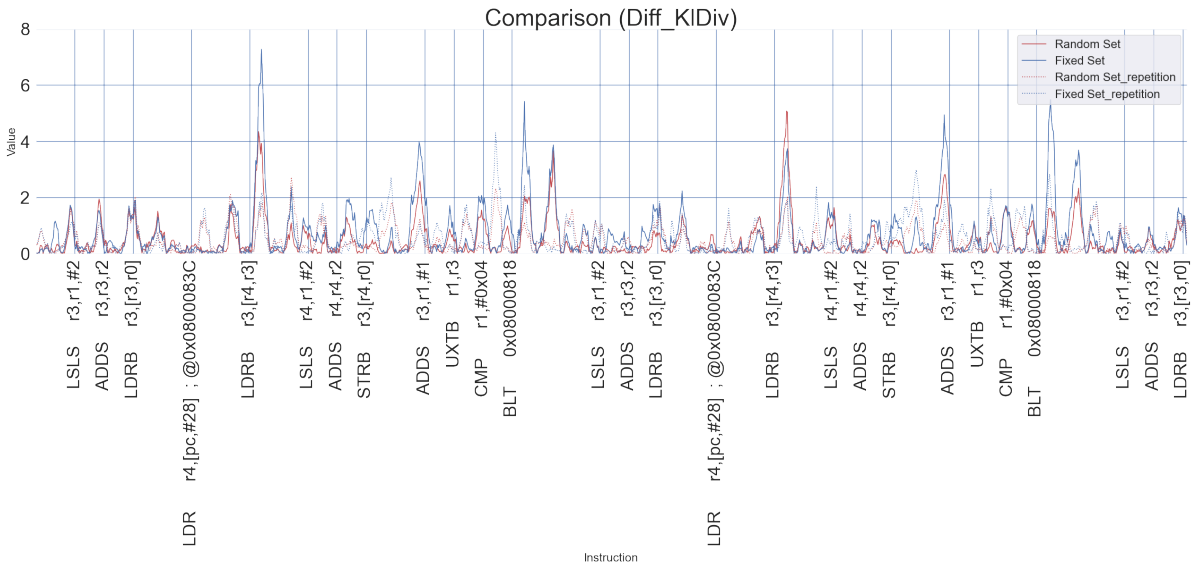


Figure 6.3: The result of specific testing on traces from STM board A with data 1 (STM_A_key1_D1_1/2) and data 2 (STM_A_key1_D1_1/2). Fixed vs fixed results shown in blue, random vs random results shown in red. Repetition sets results with dotted lines

the intensity of peaks shown by fixed set vs random, the dependency of input data on the differences can also be observed. FOr the `LSLS`, `ADDS`, `LDRB` instructions in the beginning of trace, the peaks shown by fixed and random set have same amplitudes. Moving on to `LDRB r3,[r4,r3]` up until `BLT` instruction we observe that peaks of fixed sets are higher than random sets, this section co-responds to the region where correct key data leaks.

6.2. NRF inter-board comparison

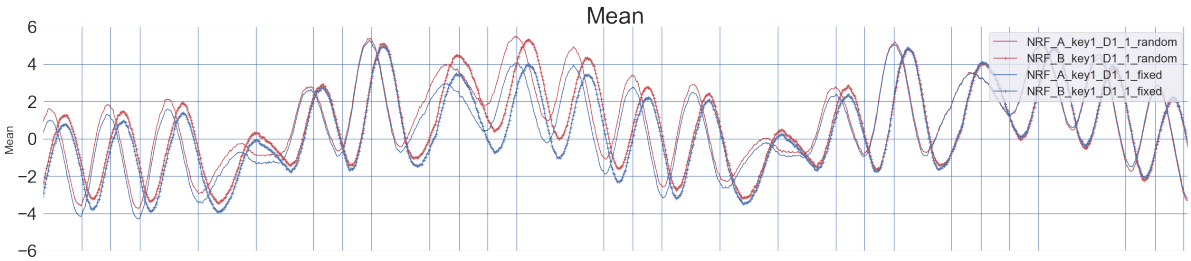
In this section AES implementation on two separate NRF board iterations marked as NRF-A and NRF-B is compared. The devices are prepared with similar hardware modifications and flashed with same binary, keeping the key and input data parameters to be the same. For presenting the results of STM inter-board comparison we will be using `STM_A_key1_D1_1/2` and `STM_B_key1_D2_1/2` (shown with marker) trace-sets. This section will discuss the results of the power profile, followed by data leakage comparison.

6.2.1. Power profile comparison

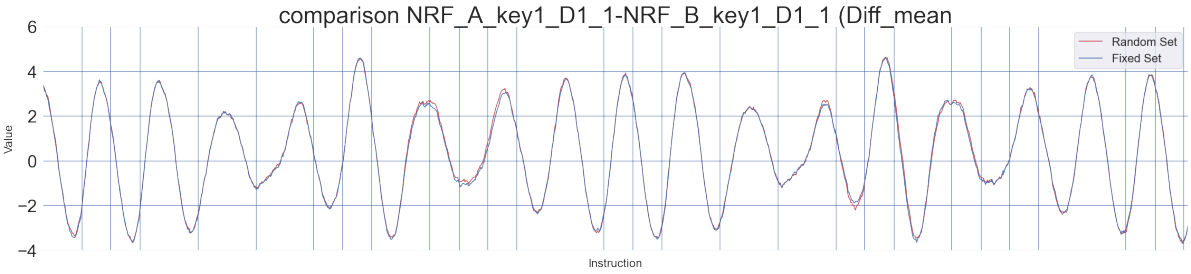
The results of the power profile comparison are presented in figure 6.4. The fixed-set (in blue) and random-set (in red) results are plotted for inter-board comparison, the inter-board sets can be distinguished by the marker. Comparison is being made between fixed-sets and random-sets from different physical boards to compare execution differences between boards.

The mean-plot of random sets and fixed-sets in 6.4a shows an identical shape, but we can observe a drift between the results from the two boards. Multiple repetitions were performed to verify this behaviour, though the reason for this drift was found to be imprecise clock period. NRF boards use a 32.768 kHz crystal as a clock source to have a low average power consumption thought, a difference was found in the clock period of two boards; this can be sourced to manufacturing process variations.

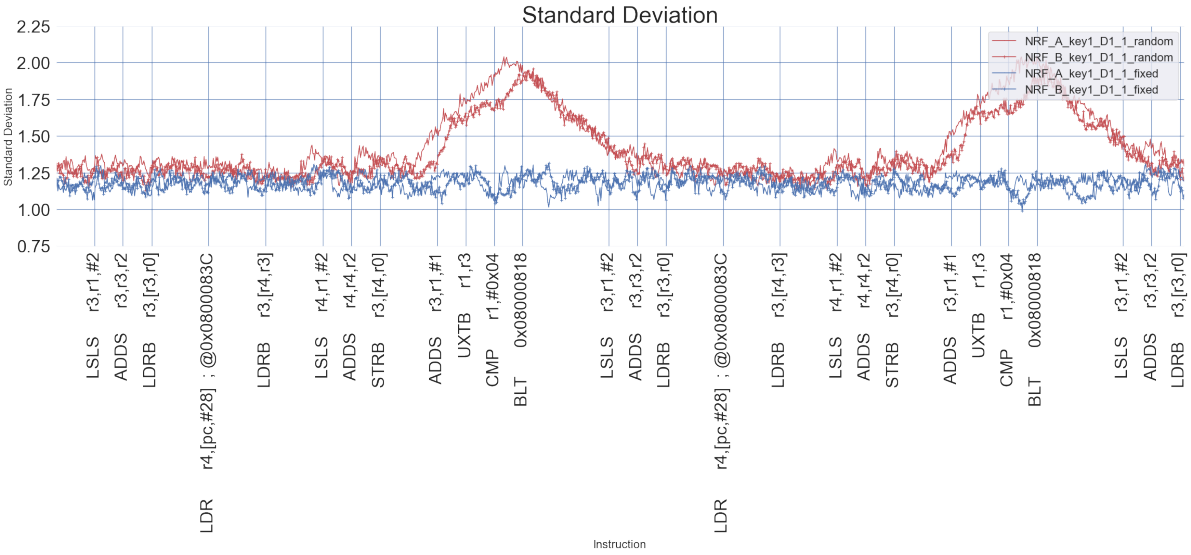
To magnify the effect of distance in means, the results for difference of means (Set B-Set A) have presented in figure 6.4b. The sections in difference of mean plots which lie above 0 imply higher power consumption by board B, while the section below 0 imply higher power consumption by board A. The difference of means plot again shows a cyclic behaviour, as it



(a) The mean execution for fixed input (in blue) and random input (in red) traces of NRF board A (NRF_A_key1_D1_1/2) and NRF board B (NRF_B_key1_D1_1/2).



(b) The mean execution for fixed input (in blue) and random input (in red) traces of NRF board A (NRF_A_key1_D1_1/2) and NRF board B (NRF_B_key1_D1_1/2).



(c) The standard deviation for fixed input and random input traces of NRF board A (NRF_A_key1_D1_1/2) and NRF board B (NRF_B_key1_D1_1/2).

Figure 6.4: Results of power profile comparison for inter-data trace-sets of NRF board A (NRF_A_key1_D1_1/2) and NRF board B (NRF_B_key1_D1_1/2). cycle-accurate instruction labels are present in the standard deviation plot

was seen in STM distance of mean plot (figure 6.1b. Looking at distance of mean plot, the relation of power consumption remains consistent across fixed-set and random-set, this shows that input data plays no role in the observed differences. The distance between the means of two sets oscillates every clock cycle, this implies a drift is present in the clock of devices which has already discussed. The shape of difference of mean plot is identical on the subsequent instructions also, so it can be determined that same behaviour is observed for repetition of instructions. This effect is observed for both random set and fixed set implying no dependency on input data, and can be contributed to the nature of physical devices.

Another observation to make is the shape of individual instruction can vary between de-

vices. The most easily identifiable difference is visible for `LDR r4, [pc,#28]` instruction, it differs considerable across both the boards yet the same trend is observe again for the subsequent `LDR r4,[pc,#28]` instruction.

In the standard deviation plots (figure 6.4c) we observe the effect of clock drift across readings from the two boards. An increase in the standard deviation value of random sets is observed after `STRB` instruction peaking at `BLT` instruction. This effect is consistent across both the boards. Variation in the clock period makes it easier to distinguish between the two boards' standard deviation plots; the differences in the curve's upwards slope vary for the two boards distinctly. These sections of trace coincide with the regions showing leakage in NRF TVLA results.

6.2.2. Data leakage comparison

TVLA, KL-Divergence and key-rank analysis are performed to comment on data leakage from inter-board sets. TVLA methodology requires performing two repetitions; the repetition set results with the same fixed inputs are also presented with a dotted line. The results for data leakage comparison on inter-data trace-sets from STM board A with fixed data 1 (STM_A_key1_D1_1/2) and fixed data 2 (STM_A_key1_D1_1/2) are presented in figure 6.5.

The leakages shown by TVLA (figure 6.5a) and KL divergence (figure 6.5b) show similar results, so we do not observe any effects that can help us distinguish between the boards. In the results of KL-divergence, we find the smaller peaks are distributed along the trace. However, we cannot distinguish the results as we find a significant variation amongst two repetitions on a similar board.

From the results of key-rank analysis on the two boards presented in figure 6.5c and 6.5d we observe that there are minor difference in the leaks of correct key data from the two boards. For NRF boards the leaks are detected in regions co-responding to sections shown by the results of KL-divergence.

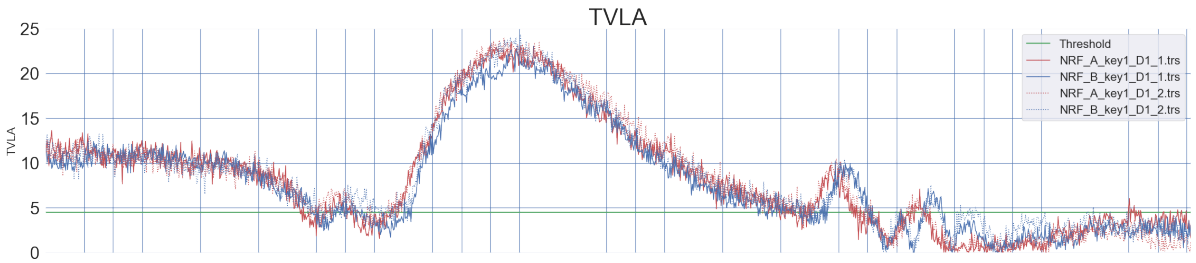
6.2.3. Comparing trace sets for differences

In this section, we perform specific testing on the trace sets, where fixed vs. fixed comparison (shown in blue) is performed using KL-Divergence to check for the sections where trace sets differ based on input data. The results of applying KL-divergence on random sets from both the trace sets are also presented in red; this gives a baseline idea of regions where we find differences in trace sets irrespective of input data.

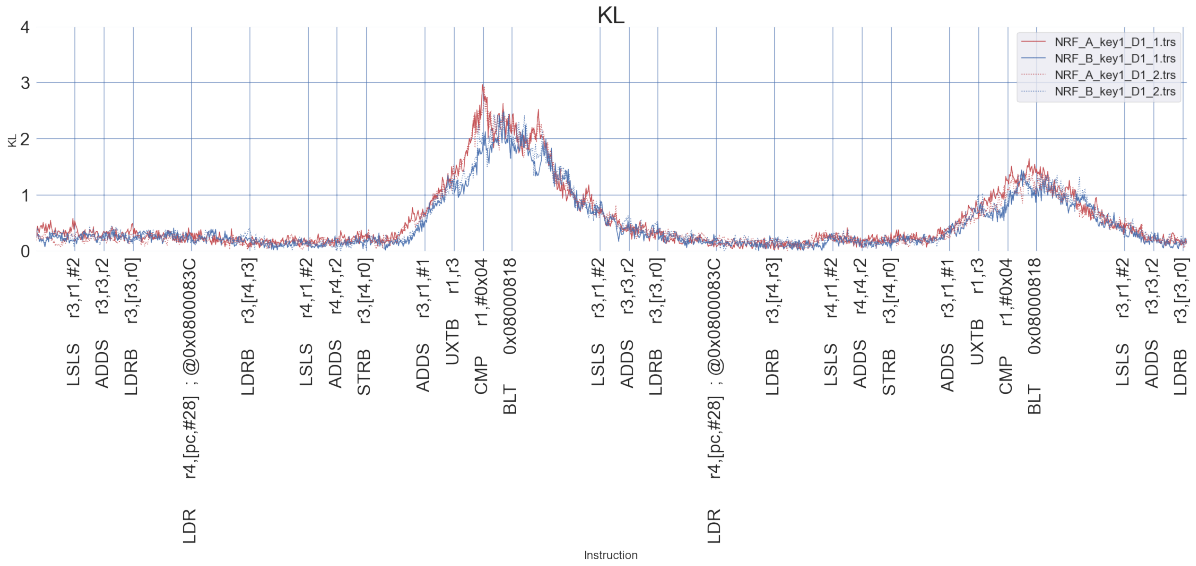
KL divergence comparison between the trace sets is presented in figure 5.6 These results show that inter-board traces have a difference in probability distribution every clock cycle. Comparing the intensity of peaks shown by fixed set vs random, the dependency of input data on the differences can also be observed. The baseline marked by random vs. random (in red) shows differences in random data sets, the fixed vs fixed comparison (in blue) has higher peaks implying more differces when fixed in put sets are compared. These results are comparable to KL-divergence on inter-board STM traces (figure 6.3).

6.3. Profiling templates

To observe the results of profiling templates across different sections of trace, a POI was chosen every 5 sample points. POI refers to point of interest, and the samples selected as POIs are used for profiling or attacking using templates. Hamming weight model was used for profiling the traces, the intermediate targeted is S-Box out. The results of profiling STM board are presented in figure 6.7 and NRF board in figure 6.9. Based on the quality of profiled templates, the type of leakage can be classified for different sections of tracef. The results of template comparison on STM board are discussed followed by NRF board.



(a) The standard deviation plot for random traces from NRF board A fixed set 1 (NRF_A_key1_D1_1/2) and fixed set 2 (NRF_A_key1_D2_1/2).



(b) The standard deviation plot for random traces from NRF board A fixed set 1 (NRF_A_key1_D1_1/2) and fixed set 2 (NRF_A_key1_D2_1/2).

(c) Key-ranking results from NRF board A fixed set 1 (NRF_A_key1_D1_1).

(d) Key-ranking results from NRF board A fixed set 2 (NRF_A_key1_D1_1).

Figure 6.5: Results of data leakage comparison for inter-board trace-sets from NRF board A fixed set 1 (NRF_A_key1_D1_1/2) and fixed set 2 (NRF_A_key1_D2_1/2).

6.3.1. STM template profiling

The results of profiling template on STM board trace is presented in figure 6.7. The model of profiled template varies over the highlighted sections, it can broadly classify the observed template into 3 models, corresponding to the leaks observed at the sample. Detail analysis of templates at different highlighted sections is presented in figure 6.8 and discussed ahead ahead.

Profiling template at the section highlighted in black, we observe that the leakage model shows a linear relation to the hamming distance of input data. The results of profiling at `BLT` instruction belonging to black section for STM A and STM B are presented in figure 6.8a and 6.8b. This type of leakage model is specific to data overwrite leaks as the power consumed to toggle bits in a register is proportional to the hamming distance of the two values. The same leakage model is profiled from the two STM boards, and the portability of attack was verified

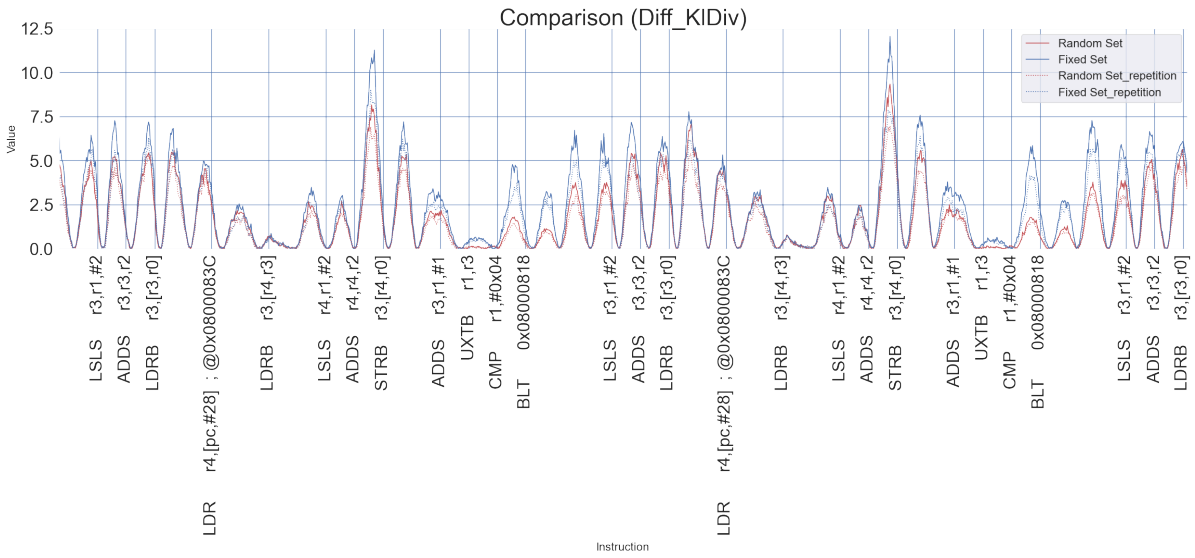


Figure 6.6: The result of specific testing on traces from NRF board A (MRF_A_key1_D1_1/2) and board B (MRF_B_key1_D1_1/2). Fixed vs fixed results shown in blue, random vs random results shown in red. Repetition sets results with dotted lines

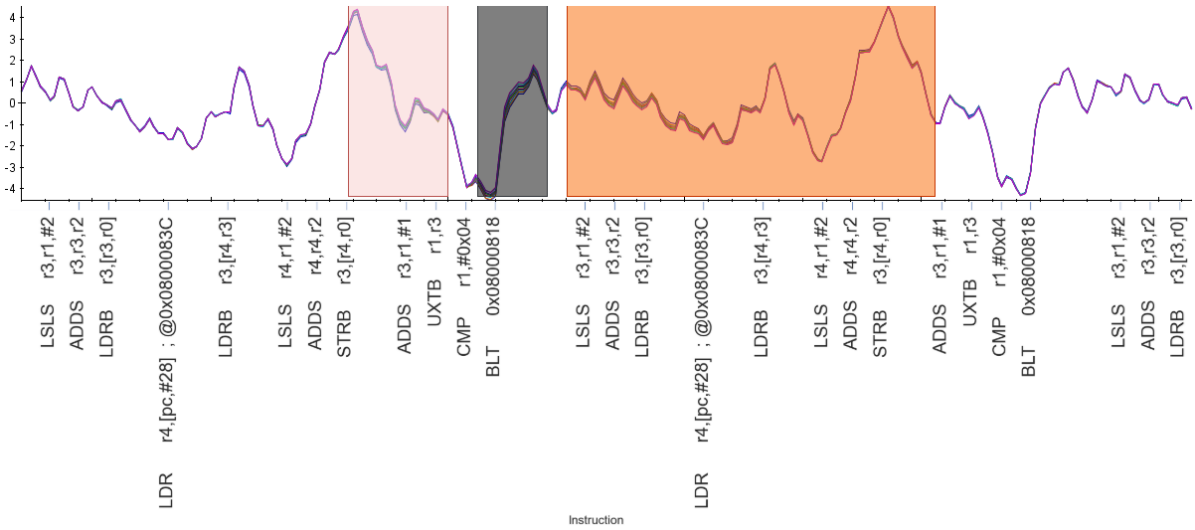
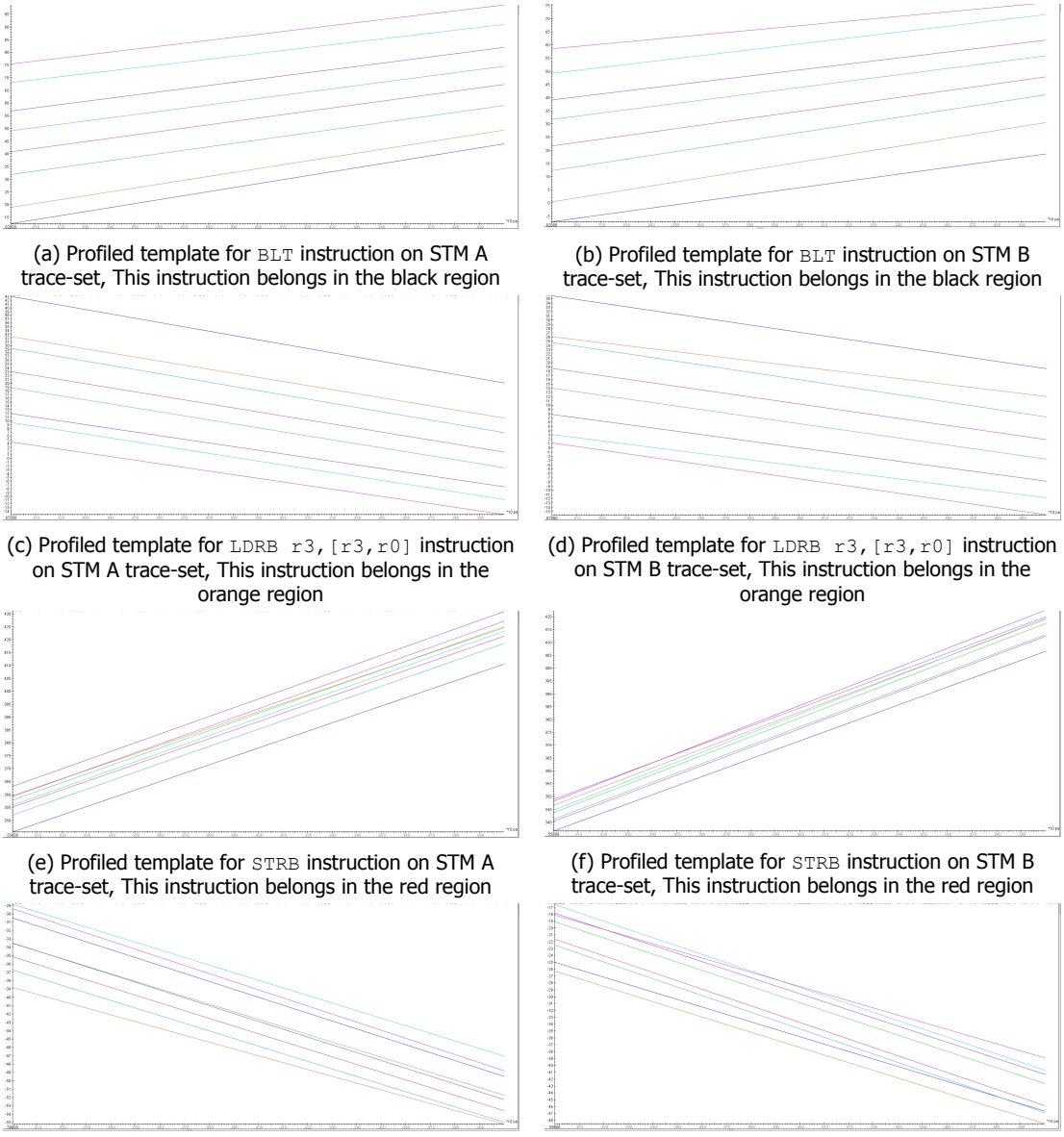


Figure 6.7: The result of profiling template on STM board A. The shape of profiled trace is similar to mean traces, though deviations can be observed in the highlighted sections. The the quality of template can be determined based on the separation of hamming weight classes. The profiled template has a different model based on the highlighted section chosen. The section highlighted in red has micro-architectural leaks, where as we observe data over write leaks in grey and orange sections

by using the template profiled on STM A to attack STM B, which successfully identified the key.

Profiling template at the section highlighted in orange, we observe that the leakage model shows an inverse relation to the hamming distance of input data. This template is the inverse of what was generated at black sections. The results of profiling at `LDRB r3,[r3,r0]` instruction belonging to orange section for STM A and STM B are presented in figure 6.8c and 6.8d. This type of leakage model is still related to data overwrite leaks as the power consumed to toggle bits in a register is proportional to the hamming distance of the two values. The same leakage model is profiled from the two STM boards, and the portability of attack was



(a) Profiled template for `BLT` instruction on STM A trace-set, This instruction belongs in the black region

(b) Profiled template for `BLT` instruction on STM B trace-set, This instruction belongs in the black region

(c) Profiled template for `LDRB r3,[r3,r0]` instruction on STM A trace-set, This instruction belongs in the orange region

(d) Profiled template for `LDRB r3,[r3,r0]` instruction on STM B trace-set, This instruction belongs in the orange region

(e) Profiled template for `STRB` instruction on STM A trace-set, This instruction belongs in the red region

(f) Profiled template for `STRB` instruction on STM B trace-set, This instruction belongs in the red region

(g) Profiled template for `UXTB` instruction on STM A trace-set, This instruction belongs in the red region

(h) Profiled template for `UXTB` instruction on STM B trace-set, This instruction belongs in the red region

Figure 6.8: Results of profiling tempalates on STM board.

verified by using the template profiled on STM A to attack STM B, which successfully identified the key.

Profiling template at the section highlighted in red, we observe that the leakage model does not show any specific pattern. The profiled model varies every clock cycle from instruction to instruction, the results of profiling model at `STRB r3,[r4,r0]` are presented in 6.8e for STM A and 6.8f for STM B. The profiled model is of lower quality in comparison to those at black and orange sections, since the plot of different hamming wight classes is close together. Moreover it is observe that the model changes based on the profiled device, template profiled on device A has differences compared to template profiled on device B. Using template profiled on STM A to attack STM B, fails in finding all the key bytes.

As we specified that the model of template changes every instruction, a template was profiled at the `UXTB` instruction also. The results of profiling model at `UXTB` instruction are

presented in 6.8g for STM A and 6.8h for STM B. It can be observed that the template differs from that profiled at `LDRB r3,[r4,r3]` instruction. There are differences in the templates profiled across the two physical devices also. Using template profiled on STM A to attack STM B, fails in finding all the key bytes.

The leaks observed in the red section can be attributed to micro-architecture leaks, when information leaks due to circuit delays and glitches. It can be observed that micro-architecture sources of leaks can vary from inter-device variations, which makes it difficult to port attacks.

6.3.2. NRF template profiling

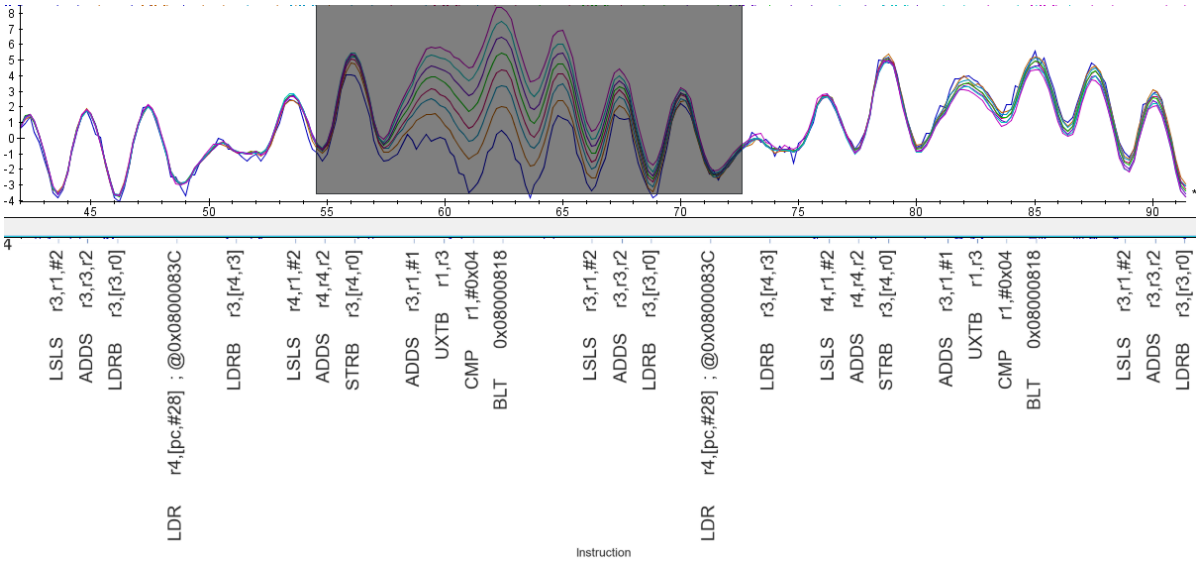
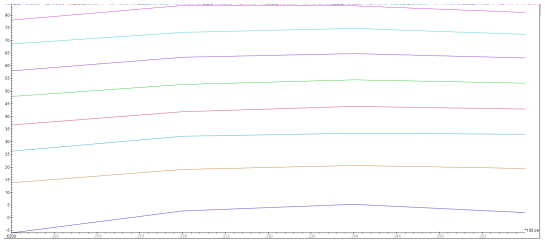
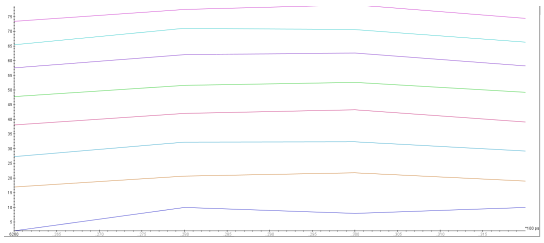


Figure 6.9: The result of profiling template on NRF board A. The shape of profiled trace is similar to mean traces, though deviations can be observed in the highlighted sections. The the quality of template can be determined based on the separation of hamming weight classes. For NRF board we only observe one type of template profile, which matches the model of data-overwrite leaks at the section highlighted in grey. This model is similar to the template generated at STM board for section highlighted grey

The results of profiling template on NRF board trace is presented in figure 6.9. The model of profiled template varies over the highlighted sections, where the well defined leakage model relative to hamming weight is observed, which is an evidence of data-overwrite leaks. We don not observe different evidence of any other sections in trace where a well defined leakage model was generated.



(a) Profiled template for `BLT` instruction on NRF A trace-set, This instruction belongs in the black region



(b) Profiled template for `BLT` instruction on NRF B trace-set, This instruction belongs in the black region

The result of template trained at `BLT` instruction on NRF board A is shown in figure 6.10a and NRF board B is shown in figure 6.10b. The relative increase in power consumption is proportional to the hamming weight in the profiled models, this model is similar to the model generated at `BLT` instruction for STM boards (figure 6.8a and 6.8b).

Profiling a template on STM board with POIs belonging to the `BLT` instruction, it was found that the template successfully attacks NRF traces to reveal the correct key. The source of leaks at the POIs is from data-overwrites, hence the profiled template is portable across boards from different manufacturers. Though if the source of leaks are from micro-architecture components, the profiled templates show significant deviations and are not portable across boards.

6.4. Summarising chapter

1. The mean power consumption for executing instructions can have minor variations across boards, this behaviour differs with instructions, is observed with repetitions and can be profiled specific to instruction.

2. The differences in clock source of two boards leads to inconsistencies in timings and synchronisation of traces, this is observed from difference of mean plots.

3. KL-Divergence can be used to expose the intensity of differences between two inter-board traces. The results of inter-board KL-divergence for the two classes shows similar behaviour.

4. We can determine the source of leaks based on the profiled template. Data-overwrite leaks have a direct relation to the hamming weight of value, Data-overwrite leaks are observed for similar section in STM and NRF trace, The profiled templates were successful in attacking traces cross device classes.

5. Micro-architecture leaks are more difficult to target for portable template attacks because the distance between the probability distributions of hamming weight classes is less and does not follows an observable pattern.

7

Inter-class comparison

In this chapter, we explore the difference in devices belonging to the the same family. The results from previous chapter have been used to point out the similarities and differences between devices of the two classes.

7.1. Inter-class comparison

This section compares trace-sets from the two classes STM and NRF, using the methodology in section 5.1. The trace-sets are chosen with the same execution parameters of key and fixed data, to maintain uniformity. Results from a repetition trace-set are plotted to present the level of variations to be expected between two repetitions.

The traces for Inter-class comparison can be `STM_X_keyY_DZ_1/2` and `NRF_X_keyY_DZ_1/2`; where parameter in green can be different, and parameter in orange is same for both trace-sets.

1. `X` can be `A` or `B` representing the board iteration, can be different for both sets;
2. `Y` can be `1` or `2` representing the encryption key, needs to be same across both sets in order to have same key data;
3. `Z` can be `1` or `2` representing the fixed set, needs to be same across both sets in order to have same input data.

Trace-sets `STM_A_key1_D1_1/2` and `NRF_A_key1_D1_1/2` are used for presenting the results of power profile comparison in 7.1.1, data leakage in 7.1.2 and verification with key-rank analysis in 7.1.3.

7.1.1. Inter-class power profile analysis

Results of performing power profile comparison on trace sets `STM_A_key1_D1_1/2` and `NRF_A_key1_D1_1/2` are presented in this section. The comparison is being performed on the section of trace highlighted in figure 5.3. The results of two classes on separate plots due to the difference in the scale of Y-axis values.

Mean traces

Fixed vs. random mean plots comparing the two devices are presented in figure 7.1. The traces from STM class have a high power consumption compared to the traces from NRF51 class; this is evident from the scale of the Y-axis. STM32 is designed for general purpose IOT applications, whereas the NRF51 is a LPE device and has a current consumption of 2mA while

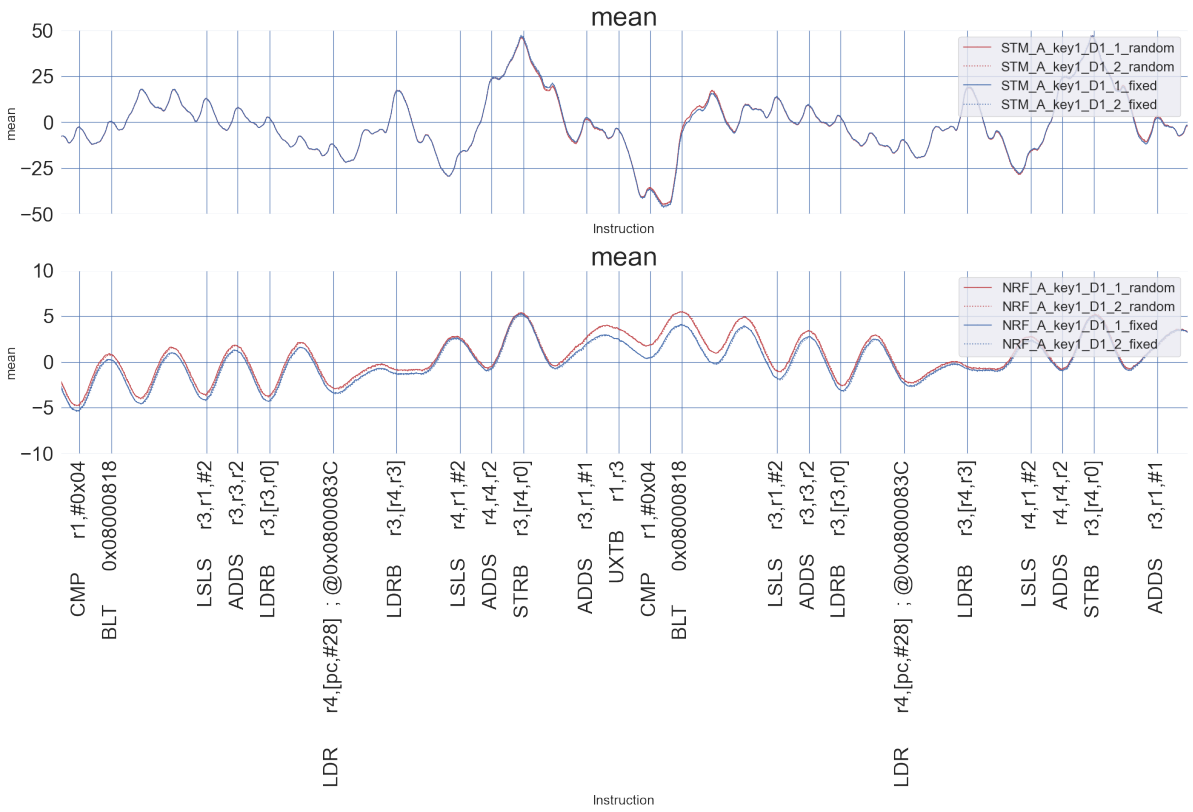


Figure 7.1: fixed vs random Mean plot for STM (top) and NRF (bottom). The Y-axis shows the power consumption, X axis represents time. The assembly instruction labels in x-axis represent the instruction being executed at the instant.

executing encryption. The low power of NRF device makes it more sensitive to noise. We observe a sinusoidal trend for NRF execution trace due to a low signal to noise ratio in the acquisition.

Cortex M0 has a three-stage pipeline, which means there can be up to 3 instructions implemented in the fetch, decode, and execute stages of the pipeline. Memory operations are performed by micro-architecture peripherals reading and writing on data-in/out registers to accesses the data pointed by address-bus in memory, while the processor core can execute other operations. Memory access takes time and greatly impacts the dynamic power, so the effect of memory instructions is significant and can be diffused to be visible while other instructions are executed.

The instruction labels on the x-axis list the instruction being executed at the clock cycle, though power trace consists of cumulative effect from all pipeline stages of the processor. Incorporating the pipeline stages and instruction scheduling into power models can improve their quality. Relating the observed effects in power traces to instructions is not easy since the source instruction of leaks can be from any pipeline stage.

On porting a code disassembly to a device with similar hardware architecture, if ported correctly, the grouping of instruction in pipeline stages will also be similar for the two classes of devices. The magnitude/contribution of leaks from different pipeline stages can vary for the two device classes, though in this case the signal from NRF device has low SNR to make solid conclusions against NRF device.

In mean-plot of power traces 7.1, results of STM class shows the power profile of each instruction cycle clearly,Repeating artifacts are observed for `LDR` and `STR` instructions in power

trace. In NRF results, the effects of individual instructions are not prominent and difficult to distinguish from the sinusoidal trend visually, this is due to the low signal to noise ratio. On execution of `LDR r4,[pc,#28` instructions, the power signal deviates from the sinusoidal trend and becomes flat. For `STRB r3,[r4,r0]` instruction also, a slack in slope of sinusoidal signal is observed. These fingerprints can vary across physical board iterations (discussed in inter-board comparison). This shows load and stores significantly affect the power consumption compared to other instructions that do not influence the sinusoidal signal in power traces.

From the mean plots of fixed (blue) vs. random (red) execution for both the device classes, we observe both the sets follow the same trend though they differ along certain sections of the traces. In the STM-mean plot, the deviation between the random and fixed sets is visible only following the `STRB r3,[r4,r0]` instruction up until the `BLT` branch less than instruction. In case of NRF plots, we can distinctly see the execution trace of fixed as well as random set; the interesting observation is that the distance between the mean of two sets increases substantially after the `STRB r3,[r4,r0]` instruction and then slowly decreases up until the `BLT` instruction. From the results of previous chapters we know these are the locations where leaks are observed. The software implementation seems to show evidence of data overwrite leaks at similar section across devices of both class.

On comparing the mean-traces plots from both families of devices, we observe the fixed vs. random lines deviate at the same power trace sections. The deviations reveal sections of code with a component of dynamic power. If the underlying data is changed for the code section, we will observe fluctuations in the specific section of power traces. A comparison of fixed to random trace-set makes it easy to identify the sections contributing to leaks.

To report results with confidence we present repetition trace-sets to verify that the observations are not incidental and can be repeated. We distinguish between the two repetitions by presenting results with a solid and a dotted line in the power profile comparison plots. To distinguish between the fixed set and random set results, we plot the results with blue and red lines, respectively.

Standard Deviation

The data of operands influences the power consumption due to the toggling of bits in peripherals when new data is loaded; these fluctuations can provide information on underlying data hence it leaks. An increase in the standard deviation of random set is observed where instructions have depend on the underlying data. The standard deviation of fixed set provides us with the base level to be expected for executing a set of operations with consistent data. It was shown from the results of inter-key comparison that the key data also plays a role in influencing the observed standard deviation.

Standard deviation plots in figure 7.2 shows the standard deviation for the fixed set consistently varies every clock cycle. This behavior is shown by the fixed sets from both the boards and repetitions. While the increase in standard deviation of random sets provides evidence of leaks, which are observed at similar trace sections for devices of two classes.

For the STM traces the deviation in random vs fixed plot occurs near the `LDRB r3,[r4,r0],STRB r3,[r4,r0]` and `BLT 0x08000818` instructions where variance of random-set is visibly higher in comparison to the fixed-set. In the case of NRF boards, the variance of a random set is higher to the fixed set for all sections of the trace. This can be used to distinguish fixed-set from random-set for NRF class. Following the execution of `STRB r3,[r4,r0]` instruction the variance of random set increasese until the `CMP r1,#0x04` instruction where it peaks and goes down until `LSLS` instruction where operation on next byte start.

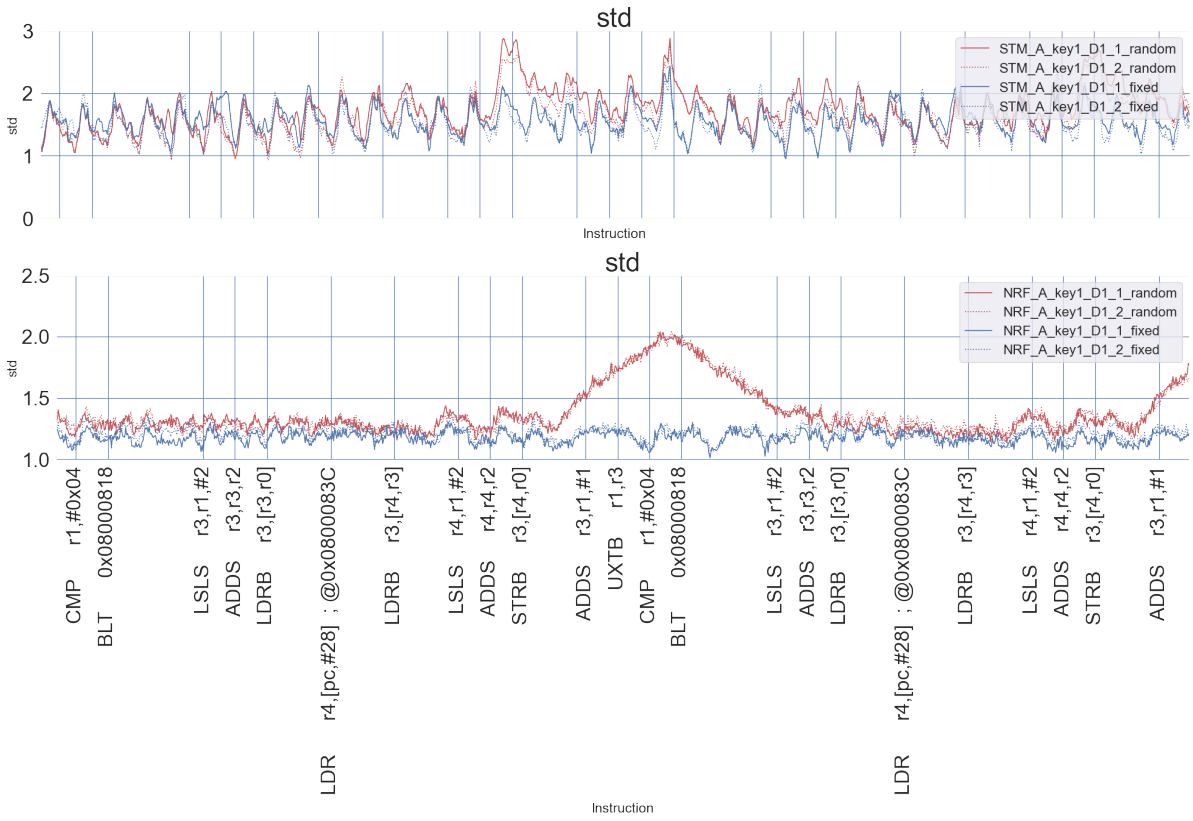


Figure 7.2: The first round sbox operation on first byte data has been selected for analysis, highlighted in the above traces with red. The raw trace from the STM board is presented above and the raw power trace from NRF board is presented below

Skew

The measure of skew provides us with information about the symmetry of the underlying probability distribution of samples. As discussed in Advanced Tools for Side-Channel Leakage Estimation [47], there is a possibility to observe leaks when higher orders statistical tools are applied. Measure of skew is used to check for leaks of a higher order. The results of skew calculation are presented in figure 7.3.

On comparing skew plots from both the families, we observe that the skew for NRF devices doesn't fluctuate a lot, and the readings stay close to 0. Whereas in the case of STM device, the skew fluctuates between the values of +-10 in certain sections of trace with low fluctuations in some regions. The skew values differ significantly between the two repetition sets, presented by a solid and dotted line. On comparing the plots of fixed vs. random sets, we do not see any interesting patterns that provide insights regarding leaks. The same behavior was observed across both the families of devices and across multiple experiments; hence skew comparison has been excluded from the results presented ahead.

7.1.2. Data leakage comparison

In this section, we present the result of applying leakage detection methodologies on the inter-class traces and compare instructions leaking intermediate data. This is done to answer the question of leaking key intermediate across two classes of boards.

Using TVLA methodology

The TVLA results for STM and NRF are presented at the top of figure 7.4 using red and blue lines respectively. The TVLA methodology specifies performing repetitions to verify results since due

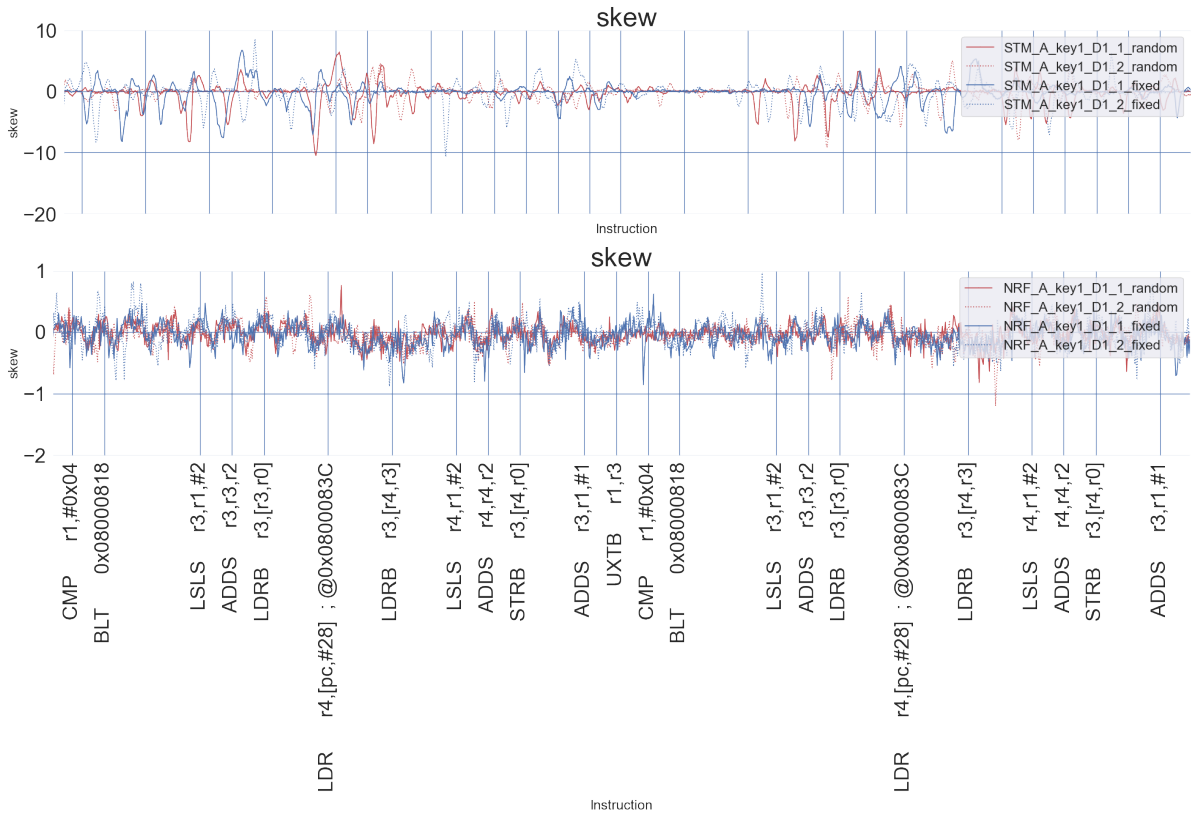


Figure 7.3: The first round sbox operation on first byte data has been selected for analysis, highlighted in the above traces with red. The raw trace from the STM board is presented above and the raw power trace from NRF board is presented below

to the statistical nature of the test, minor deviations can occur between two repetitions; the two repetitions are shown by a solid and a dashed line.

The TVLA results for the two classes differ significantly. For STM board,the TVLA value rises above threshold at `STRB r3,[r4,r0]` instruction; goes down at `UXTB` instruction and rises again covering `CMP r1,#0x04` and `BLT` instructions. For the NRF board, the TVLA results are higher than the threshold value of 4.5 (represented by a green line) for most of the plot; following TVLA methodology, this implies all instructions are leaky. A curve starting at `STRB r3,[r4,r0]` instruction which peaks at `CMP r1,#0x04` instruction is observed on NRF board. As discussd in chapters before the TVLA results need to be considered with caution as they are greatly influenced by the choice of input set. Using TVLA results, both device classes show leaks at different instructions.

TVLA methodology identifies the sections of trace which have data leaks, though TVLA peaks do not imply leaks of correct intermediate/key data. It might not be the correct key-data, but it can provide information to an attacker, which helps break the implementation. The correct key data may not leak directly, the leaked intermediate values can be plugged into mathematical equations to deduce the secret-key data.

Using KL-Divergence

KL-divergence is a measure of difference between two distributions. The results of calculating $KL_{(f|r)}$ for STM and NRF trace-sets are presented in the bottom plot of figure 7.4 with red and blue color respectively. The results of $KL_{(f|r)}$ show a trend similar to the results of $TVLA_{f|r}$, with some additional minor peaks.

The results of KL-divergence of traces of two classes gives significantly different results.

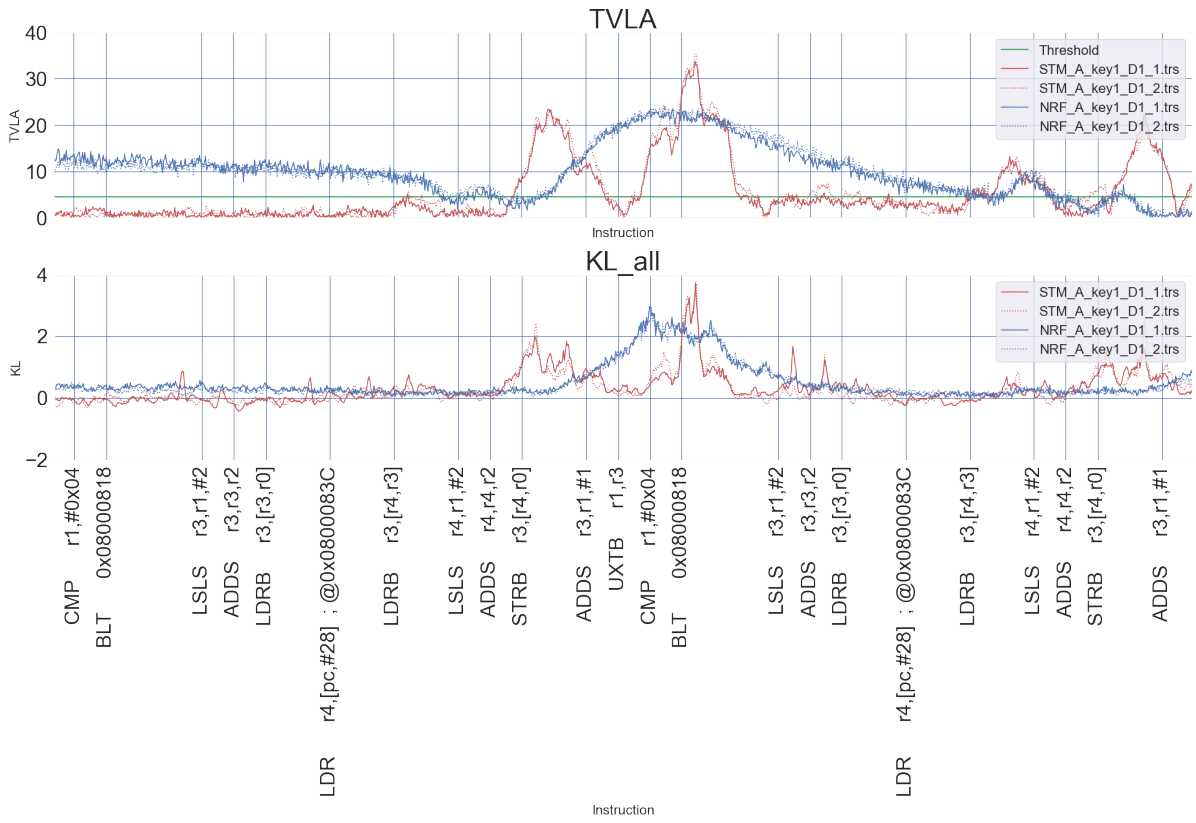


Figure 7.4: The first round sbox operation on first byte data has been selected for analysis, highlighted in the above traces with red. The raw trace from the STM board is presented above and the raw power trace from NRF board is presented below

Though we it is known from the quality of profiling templates in section 6.3, that these results to coincide with section where a high quality template was profiled.

7.1.3. Key-rank analysis

Key-rank analysis was performed using the Riscure inspector software using the known key analysis module. The known key analysis is used to determine the probability of key data leaking along the trace.

The key-rank analysis results have been added as a transparent layer on top of the TVLA and KL divergence plots for both boards. The **Red** regions are the index locations where the correct key was ranked first and it was ranked low in **Blue** regions.

The results of applying key-rank analysis on 100 and 1.2k random-trace sets, are shown to present the evolution of results with an increasing number of traces. The TVLA and KL-divergence results, which were calculated with 1.2k traces, are compared against the key-rank analysis done with 1.2k traces to have a valid comparison.

STM key-rank analysis

The key-rank analysis results using 100 traces have been overlaid on TVLA, and KL divergence results in 7.5. The correct key byte starts leaking at `ADDS r4,r4,r2` instruction, which is just a clock cycle before `STRB r3,[r4,r0]` instruction. We think this is an effect from the three-stage pipeline, while the `ADDS` instruction is being executed; the data is being pre-fetched for the `STRB` instruction, which is being observed as leak. The leak of key-data stops at `ADDS r3,r3,r2` instruction subsequent to which a `LDRB r3,[r3,r0]` instruction is executed. The pre-fetch of `LDRB` instruction start while `ADDS` instruction is being executed.

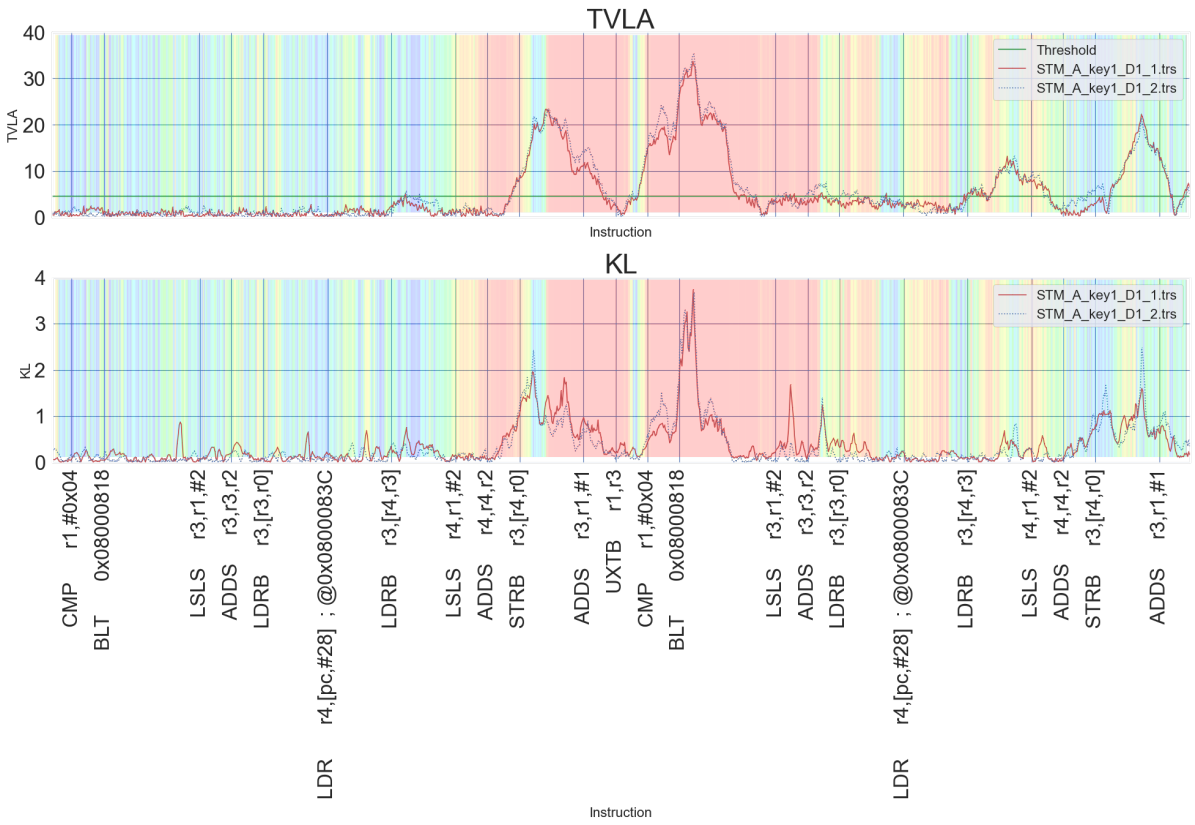


Figure 7.5: Key-rank analysis results overlaid on TVLA and KL-divergence calculations from STM board for 100 traces. The colour overlay depicts the probability of key data being leaked along the trace, red colour marking strong leaks and blue marking weak leaks

The substituted intermediate, which is in R3, is saved in memory by `STRB` instruction. To do so, the data is loaded onto B Bus and dataOut register of processor core (figure 2.2). The subsequent `ADDS` instruction overwrites the value in R3 with the index of subsequent byte, so key-related data may leak when overwriting. In our traces, the correct key byte leaks in the subsequent instructions even though no operations are being performed directly on key data. In the analyzed s-box implementation, the loop operates on 4 bytes, four times to operate on total 16 bytes of data; check for the loop occurs at `cmp r1,#0x04` instruction. The check compares the relative value in R1 to #0x04 and branches in the next instruction because R1 value is less than 4. The subsequent instructions `LSLS` and `ADDS` compute the relative index from which the next key data is to be loaded by `LDRB`, which is when leak of key data stops. We find this to be an interesting behavior since the data stops leaking when data in memory bus A will be overwritten by new data, as discussed in the microprocessor subsystem implementation 2.1.4.

There are gaps in resulting leaks at `STRB` and `UXTB` instructions; we do not have an explaination for these effects.

The results from 1.2k traces are shown in figure 7.6. The correct key data starts leaking from the `LDRB r3, [r4,r3]` instruction until the `STRB r3,[r4,r0]` instruction. There are sections of trace towards the right side where the correct key is not ranked highly (represented in yellow); the program has already started operations on the second key byte. The reasoning for observing leaks of first key byte data is that it is being overwritten in the DataOut register, which only occurs when subsequent store instructions are called.

The reasoning for correct key data leaking on `LDRB r3,[r4,r3]` instruction can be due

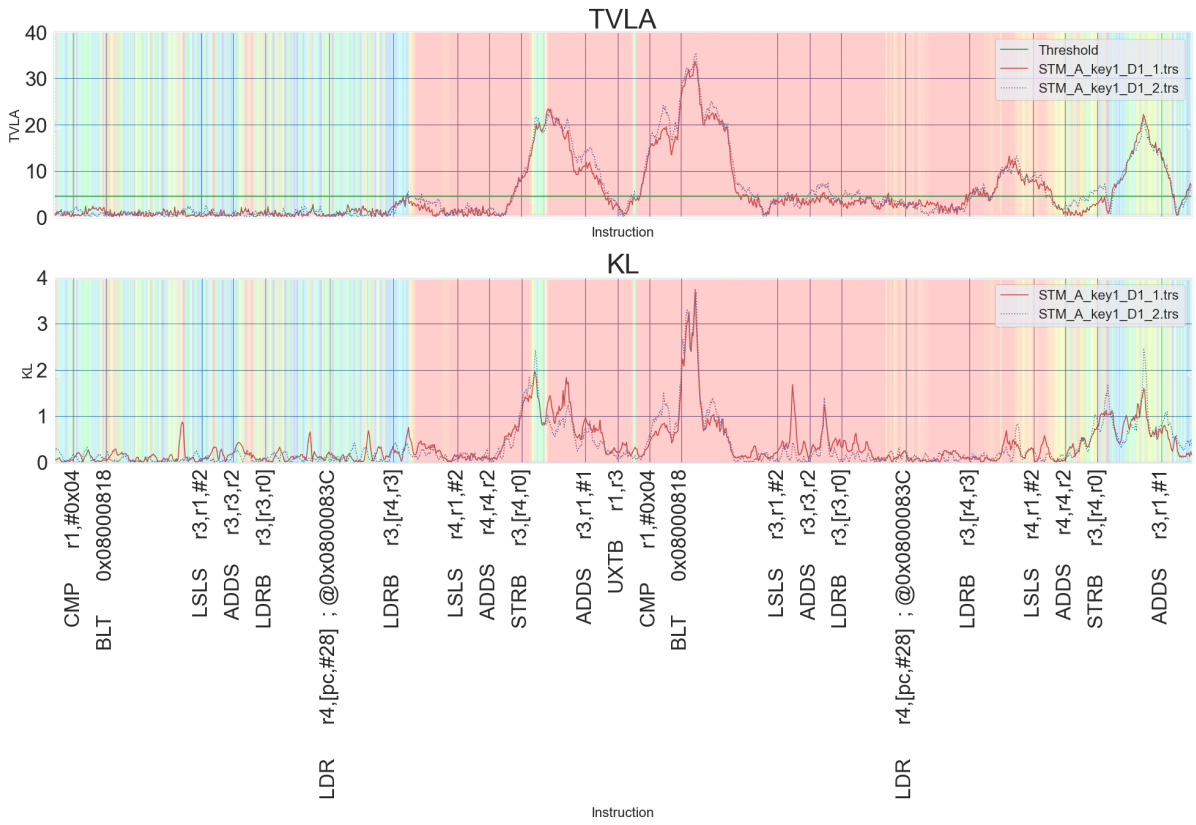


Figure 7.6: Key-rank analysis results overlaid on TVLA and KL-divergence calculations from STM board for 1.2k traces. The colour overlay depicts the probability of key data being leaked along the trace, red colour marking strong leaks and blue marking weak leaks

to key data being loaded on the bus. The key byte also leaks while the arithmetic instructions are being performed on it, and it keeps leaking until the S-box function loads the next key byte for operation at subsequent `LDRB r3,[r4,r3]` instruction.

The section of arithmetic instructions can be classified to be highly leaky. The correct key is found just using 100 traces compared to Load instruction, where the correct key is ranked correctly after 1.2k traces. The behavior also depends on the instructions being executed before and after these.

For comparison we have presented the key-rank evolution plot for `LDRB r3,[r3,r0]`, from `LDRB r3,[r4,r3]` and `LSLS r4,r1,#2` in figure 2.8. The correct key byte was retrieved from `LSLS r4,r1,#2` instruction in less than 100 traces, in case of `LDRB r3,[r4,r3]` instruction we can see the rank of correct key reduce slowly and at 1200 traces it converges to rank 1. Whereas in the case of `LDRB r3,[r3,r0]`, the correct key was not ranked one even after 1200 traces, and the plot doesn't seem to converge. More on key-rank evolution is presented in **??**.

NRF key-rank analysis

Key-rank analysis on 100 traces is resented in figure 7.7. The correct key byte starts leaking at `STRB r3,[r4,r0]` instruction, and leaks until `ADDS r3,r3,r2` instruction. NRF results differs from the key-rank results on STM boards 7.5 showing additional leaks at arithmetic (`LSLS`) instructions, NRF and STM results show a similar trend subsequent to `STRB` instruction.

The leaks observed in NRF board seem to have a strong effect on dynamic power, and its effect is diffused showing up while other instructions are being executed. We can infer that

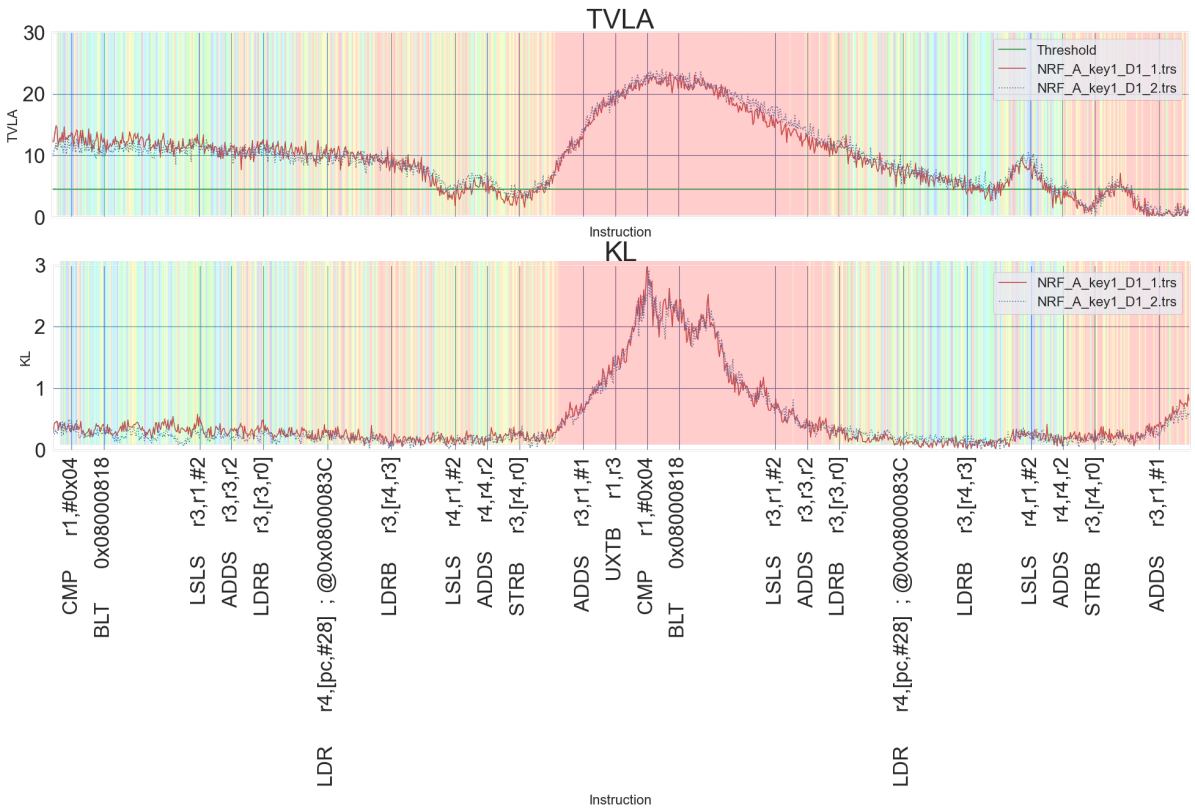


Figure 7.7: Key-rank analysis results overlaid on TVLA and KL-divergence calculations from NRF board for 100 traces. The colour overlay depicts the probability of key data being leaked along the trace, red colour marking strong leaks and blue marking weak leaks

the correct key byte is on the bus after the `STRB` instruction, which leaks over the subsequent instructions due to data-overwrite.

The key-rank results for NRF board with 1.2k traces are presented in figure 7.8. The coverage of the red region increases to include the `LDRB r3,[r3,r0]` also along with the arithmetic instructions.

The results for TVLA and KL divergence are being supported by the key-rank analysis as the red region coincides with the peaks. Another interesting behavior observed in NRF boards is the leak of the fist key byte, when the operations are being performed on the next byte of data. This leak is observed due to register-overwrites from `LDR` instruction.

7.2. Summarising chapter

In this chapter, we compared the behaviour of boards from two different manufacturers when running the same implementation.

1. Power profile of boards of different classes differ completely yet we observe leaks at similar trace sections (`CMP` and `BLT` instruction), with similar profiled template (linear hamming distance).

2. Inter-class devices show similar leaks due to data-overwrite in pipeline registers, the fix has to be applied at assembly level to clear registers before reuse.

3. Evidence of micro-architecture leaks is observed from STM devices at arithmetic instruction (specifically `CMP`). In case of NRF device only register overwrite leaks are observed

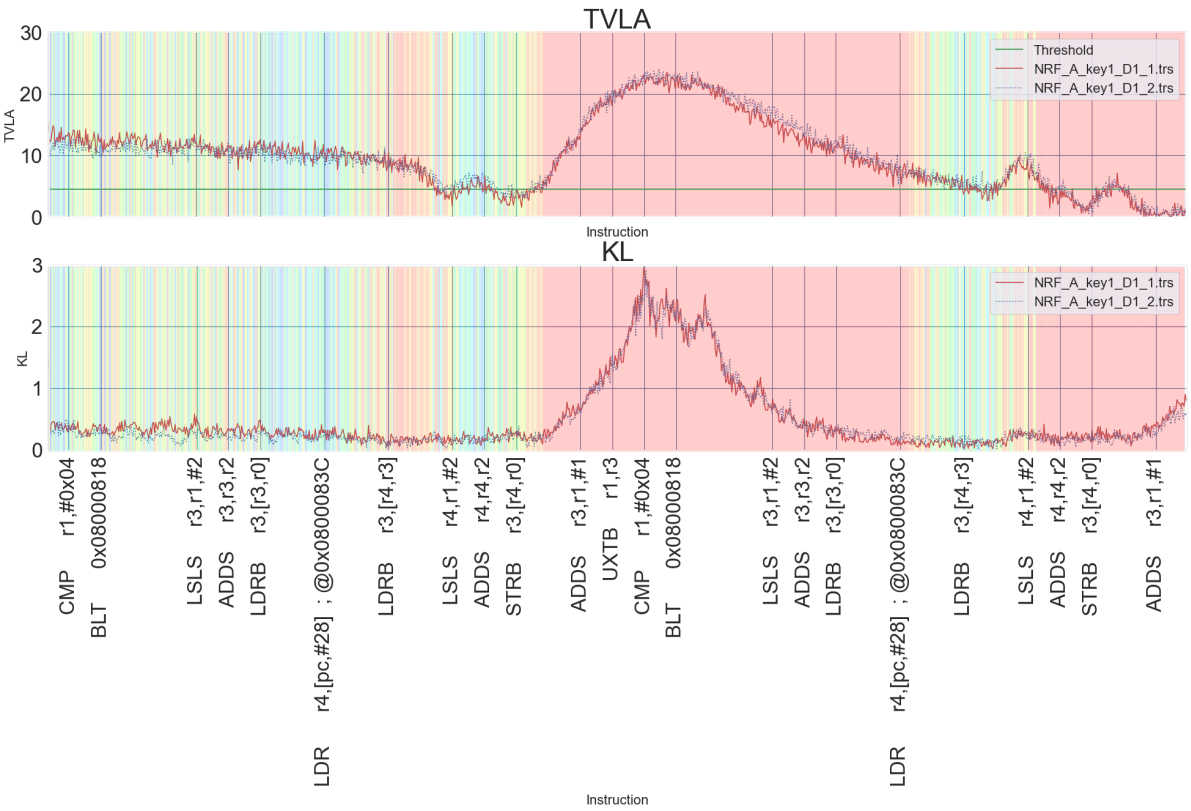


Figure 7.8: Key-rank analysis results overlaid on TVLA and KL-divergence calculations from STM board for 1.2k traces. The colour overlay depicts the probability of key data being leaked along the trace, red colour marking strong leaks and blue marking weak leaks

over a wide section of trace, we assume the intensity of data overwrite leaks is strong and masks the evidence of micro-architecture leaks that can be present.

4. TVLA and KL-divergence are useful in knowing the localised trace-section to be targeted to check for leaks, though the observed peaks for these measures might not relate to the intensity of data leaks.

8

Conclusion

This chapter compiles the results presented in previous sections of the report and uses them to answer the research questions posed. We end by presenting the contributions, limitations, and future work for this project.

8.1. Research questions

In chapter 5 we discuss the similarities and dissimilarities of traces from the same physical board on varying input and key data to answer the research question.

Question A How significant are the inter-key and inter-data variations?

Based on our experiments we conclude that there are significant differences in the trace sets acquired from the same physical board for varying input and key data. The results of TVLA methodology are greatly influenced by the choice of input data, where as KL-divergence gives more uniform results and is less influenced by input and key data. The effect of key data in influencing standard deviation was observed in boards of both classes.

KL-divergence was used for specific tests on, (fixed vs fixed) comparisons to detect the instructions where probability distribution is affected. We do not observe explainable effects on NRF board data, though for STM board we notice the effect of changing data influences is more significant than changing key data. Changing key data has an effect on instruction manipulating with key data which include arithmetic instructions and store, Changing input data has an effect on instruction loading, performing arithmetic and storing.

In chapter 6 we have focused on comparing traces from different physical boards to answer the research question.

Question B How significant are inter-device manufacturing variations?

The analysis of inter-board traces reveals there are significant differences at the instruction level. A drift in clock and micro-architecture variations are the source of these differences. Comparing means across the physical board iterations shows that execution of instructions differs in a systematic manner specific to each type of instruction; a repetitive behavior is observed for subsequent occurrences of each instruction which confirms our method of measuring differences gives consistent results.

Applying KL-divergence on for specific test on inter board traces, tells that underlying PDF differs occur every cycle specific to the instruction being executed. Results profiled for specific instruction can be used to identify the level of inter-board variations.

Based on the source of the leak, it is possible to profile templates portable across physical boards when POIs chosen have evidence of data-overwrite leaks, this was shown by porting template profiled on STM traces to attack NRF traces. While templates profiled to exploit micro-architectural leaks are influenced by inter-device variations, making them less suitable to be ported.

Chapter 7 we have profiled the traces from STM and NRF board to compare leakages across boards from the same class.

> **Question C** What is the impact of micro-architectural differences across manufacturers, on the SCA leakage for a given software implementation?

Devices belonging to the same family but from different manufacturers (class) can have significant variations in terms of power profile comparison. Yet if the software implementation does not incorporates clearing of registers and masking known value leaks, the two classes can leak in a similar manner. The trace (instruction) sections with data-overwrite leaks are similar for boards of the two classes. We present the porting of template attacks across boards of the two classes by exploiting data-overwrite leaks.

. The results of micro-architecture leaks in STM class were evolved in inter-board comparison. Micro-architecture leaks could not be compared across devices of the two classes since we were not able to detect micro-architecture leaks for NRF class due to low SNR. We show the viability of our methodology for analyzing software implementation across boards from the same class.

8.2. Contributions

Our work contributes towards the following:

- Presenting a methodology for comparing software implementation across devices of the same class (with the same instruction set), to comment on the resilience of micro-architecture designs against leaks.
- KL-Divergence is used to quantify the differences across trace-sets, the peaks of these results coincide with the locations where leaks are observed implying its viability in checking for leaks.
- The influence of key and input data on the results of leakage detection methodologies is shown.
- Templates have been profiled at different sections of power trace and used to comment on the source of leaks based on the generated template and the instructions being executed. Portability of these templates based on the source of leaks is discussed.
- The results show that source of leaks in the physical layer can vary for devices from different manufactures. In order to comment confidently on the security of a portable software implementation it is necessary to perform independent device evaluations.
- Creating a data-set of synchronized traces from STM32f0 and NRF51 devices.

8.3. Limitations

A limitation of our work is that we are unable to provide the reasoning for the observed similarities and dissimilarities between the target devices because the RTL schematics are not publicly available. The source of leaks being caused by micro-architecture or register-overwrite is distinguished based on the profiled template. We assume an ideal behaviour of flip flops used to implement the register, which leads to a linear relation between hamming distance classes in the profiled template.

Another limitation of our work is that we use an unsecured AES implementation to compare the leaks from different devices. This provides a good starting point for comparing devices, though with no counter-measures both the devices show strong leaks. The use of counter-measures in software implementations and their portability has not been evaluated in this work. The leaks are observed not only from the micro-architecture sources but also register over-writes and known value leaks, which can be mitigated in a secure implementation.

8.4. Future work

The target devices used for evaluating our methodology differed in functional specification, STM32 is a general purpose device and NRF51 is low power device meant for Bluetooth radios. The target devices from same instruction but different manufacturer set were chosen to check if we observe similar leaks. By performing the same evaluation between the different series of STM32 boards that vary in memory specification, the influence of memory size and memory technology on leaks can be observed. Since memory instruction loading and storing data contribute highly to leaks.

This work can be expanded further by comparing secure software implementations with counter-measures to evaluate the portability of countermeasures across devices. Since the devices from different manufacturers vary in the RTL implementation of instruction sets, it can be interesting to see if it impacts the efficiency of countermeasures.

Lastly there are not many devices with publicly available RTL schematic, but being able to link the observed leaks to sources at RTL level can be useful for the field to mitigate leaks at their source.

Bibliography

[1] P. Kim, D. Han, and K. C. Jeong, *Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to aes and sha-2,* Quantum Information Processing **17**, 339 (2018).

[2] T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. M. Shalmani, *On the power of power analysis in the real world: A complete break of the keeloq code hopping scheme,* in *Annual International Cryptology Conference* (Springer, 2008) pp. 203–220.

[3] A. Moradi, M. Kasper, and C. Paar, *Black-box side-channel attacks highlight the importance of countermeasures,* in *Cryptographers' Track at the RSA Conference* (Springer, 2012) pp. 1–18.

[4] D. Oswald and C. Paar, *Breaking mifare desfire mf3icd40: Power analysis and templates in the real world,* in *International Workshop on Cryptographic Hardware and Embedded Systems* (Springer, 2011) pp. 207–222.

[5] P. C. Kocher, *Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems,* in *Annual International Cryptology Conference* (Springer, 1996) pp. 104–113.

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

[7] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, *Towards sound approaches to counteract power-analysis attacks,* in *Advances in Cryptology — CRYPTO' 99*, edited by M. Wiener (Springer Berlin Heidelberg, Berlin, Heidelberg, 1999) pp. 398–412.

[8] L. Goubin and J. Patarin, *Des and differential power analysis the "duplication" method,* in *International Workshop on Cryptographic Hardware and Embedded Systems* (Springer, 1999) pp. 158–172.

[9] E. Trichina, T. Korkishko, and K. H. Lee, *Small size, low power, side channel-immune aes coprocessor: design and synthesis results,* in *International Conference on Advanced Encryption Standard* (Springer, 2004) pp. 113–127.

[10] M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre, *A formal study of power variability issues and side-channel attacks for nanoscale devices,* in *Advances in Cryptology – EUROCRYPT 2011*, edited by K. G. Paterson (Springer Berlin Heidelberg, Berlin, Heidelberg, 2011) pp. 109–128.

[11] C. Whitnall, E. Oswald, and F.-X. Standaert, *The myth of generic dpa... and the magic of learning,* in *Cryptographers' Track at the RSA Conference* (Springer, 2014) pp. 183–205.

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

[13] D. McCann, C. Whitnall, and E. Oswald, *Elmo: Emulating leaks for the arm cortex-m0 without access to a side channel lab.* IACR Cryptol. ePrint Arch. **2016**, 517 (2016).

[14] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, *X-deepsca: Cross-device deep learning side channel attack*,* in *2019 56th ACM/IEEE Design Automation Conference (DAC)* (2019) pp. 1–6.

[15] J. Bungo, *Arm cortex-m0 designstart processor and v6-m architecture,* ARM. url: httpXG-GwwwFs seF omF rGPHIPGfilesGPHIPGHWGwH vTw QIPF pdf .

[16] *ARM6 microarchitectures,* https://en.wikichip.org/wiki/arm_holdings/microarchitectures/arm6, accessed: 2020-09-30.

[17] J. Daemen and V. Rijmen, *Aes proposal: Rijndael,* (1999).

[18] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, *The em side—channel (s),* in *International workshop on cryptographic hardware and embedded systems* (Springer, 2002) pp. 29–45.

[19] G. de Souza Faria and H. Y. Kim, *Differential audio analysis: a new side-channel attack on pin pads,* International Journal of Information Security **18**, 73 (2019).

[20] S. Mangard and K. Schramm, *Pinpointing the side-channel leakage of masked aes hardware implementations,* in *International Workshop on Cryptographic Hardware and Embedded Systems* (Springer, 2006) pp. 76–90.

[21] A. A. Ding, L. Zhang, F. Durvaux, F.-X. Standaert, and Y. Fei, *Towards sound and optimal leakage detection procedure,* in *International Conference on Smart Card Research and Advanced Applications* (Springer, 2017) pp. 105–122.

[22] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi, *et al.*, *A testing methodology for side-channel resistance validation,* in *NIST non-invasive attack testing workshop*, Vol. 7 (2011) pp. 115–136.

[23] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi, and S. Saab, *Test vector leakage assessment (tvla) methodology in practice (extended abstract),* (2013).

[24] C. Whitnall and E. Oswald, *A critical analysis of iso 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'),* in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, 2019) pp. 256–284.

[25] J. Park, X. Xu, Y. Jin, D. Forte, and M. Tehranipoor, *Power-based side-channel instruction-level disassembler,* in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)* (IEEE, 2018) pp. 1–6.

[26] F.-X. Standaert, T. G. Malkin, and M. Yung, *A unified framework for the analysis of side-channel key recovery attacks,* in *Annual international conference on the theory and applications of cryptographic techniques* (Springer, 2009) pp. 443–461.

[27] S. Chari, J. R. Rao, and P. Rohatgi, *Template attacks,* in *International Workshop on Cryptographic Hardware and Embedded Systems* (Springer, 2002) pp. 13–28.

[28] C. Rechberger and E. Oswald, *Practical template attacks,* in *International Workshop on Information Security Applications* (Springer, 2004) pp. 440–456.

[29] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater, *Template attacks in principal subspaces,* in *International Workshop on Cryptographic Hardware and Embedded Systems* (Springer, 2006) pp. 1–14.

[30] O. Acıiçmez, Ç. K. Koç, and J.-P. Seifert, *Predicting secret keys via branch prediction,* in *Cryptographers' Track at the RSA Conference* (Springer, 2007) pp. 225–242.

[31] O. Aciiçmez, *Yet another microarchitectural attack: exploiting i-cache,* in *Proceedings of the 2007 ACM workshop on Computer security architecture* (2007) pp. 11–18.

[32] J. Balasch, B. Gierlichs, V. Grosso, O. Reparaz, and F.-X. Standaert, *On the cost of lazy engineering for masked software implementations,* in *International Conference on Smart Card Research and Advanced Applications* (Springer, 2014) pp. 64–81.

[33] Y. Le Corre, J. Großschädl, and D. Dinu, *Micro-architectural power simulator for leakage assessment of cryptographic software on arm cortex-m3 processors,* in *International Workshop on Constructive Side-Channel Analysis and Secure Design* (Springer, 2018) pp. 82–98.

[34] M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre, *A formal study of power variability issues and side-channel attacks for nanoscale devices,* in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 2011) pp. 109–128.

[35] O. Choudary and M. G. Kuhn, *Template attacks on different devices,* in *International Workshop on Constructive Side-Channel Analysis and Secure Design* (Springer, 2014) pp. 179–198.

[36] M. Medwed and E. Oswald, *Template attacks on ecdsa,* in *International Workshop on Information Security Applications* (Springer, 2008) pp. 14–27.

[37] E. Oswald and S. Mangard, *Template attacks on masking—resistance is futile,* in *Cryptographers' Track at the RSA Conference* (Springer, 2007) pp. 243–256.

[38] M. O. Choudary and M. G. Kuhn, *Efficient, portable template attacks,* IEEE Transactions on Information Forensics and Security **13**, 490 (2017).

[39] *DAC '19: Proceedings of the 56th Annual Design Automation Conference 2019* (Association for Computing Machinery, New York, NY, USA, 2019).

[40] S. Bhasin, A. Chattopadhyay, A. Heuser, D. Jap, S. Picek, and R. R. Shrivastwa, *Mind the portability: A warriors guide through realistic profiled side-channel analysis,* Cryptology ePrint Archive, Report 2019/661 (2019), https://eprint.iacr.org/2019/661.

[41] S. Picek, A. Heuser, A. Jovic, S. A. Ludwig, S. Guilley, D. Jakobovic, and N. Mentens, *Side-channel analysis and machine learning: A practical perspective,* in *2017 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2017) pp. 4095–4102.

[42] E. De Mulder, S. Gummalla, and M. Hutter, *Protecting risc-v against side-channel attacks,* in *2019 56th ACM/IEEE Design Automation Conference (DAC)* (IEEE, 2019) pp. 1–4.

[43] M. A. Shelton, N. Samwel, L. Batina, F. Regazzoni, M. Wagner, and Y. Yarom, *Rosita: Towards automatic elimination of power-analysis leakage in ciphers,* arXiv preprint arXiv:1912.05183 (2019).

[44] A. G. Bayrak, F. Regazzoni, D. Novo, P. Brisk, F.-X. Standaert, and P. Ienne, *Automatic application of power analysis countermeasures,* IEEE Transactions on Computers **64**, 329 (2013).

[45] S. Gao, B. Marshall, D. Page, and T. Pham, *Fenl: an ise to mitigate analogue micro-architectural leakage,* IACR Transactions on Cryptographic Hardware and Embedded Systems , 73 (2020).

[46] C. Whitnall and E. Oswald, *A cautionary note regarding the usage of leakage detection tests in security evaluation,* Cryptology ePrint Archive, Report 2019/703 (2019), https://eprint.iacr.org/2019/703.

[47] T. Schneider, A. Moradi, F.-X. Standaert, and T. Güneysu, *Bridging the gap: Advanced tools for side-channel leakage estimation beyond gaussian templates and histograms,* Cryptology ePrint Archive, Report 2016/719 (2016), https://eprint.iacr.org/2016/719.