Politecnico di Torino

Cybersecurity for Embedded Systems
01UDNOV

Master's Degree in Computer Engineering

Project Title
Project Report

Candidates:
Name Surname (student ID)
Name Surname (student ID)
Name Surname (student ID)

Referee:
Prof. Paolo Prinetto

Contents

List of Figures

List of Tables

Abstract

This is the space reserved for the abstract of your report. The abstract is a summary of the report, so it is a good idea to write after all other chapters. The abstract for a thesis at PoliTO must be shorter than 3500 chars, try to be compliant with this rule (no problem for an abstract that is a lot shorter than 3500 chars, since this is not a thesis). Use short sentences, do not use over-complicated words. Try to be as clear as possible, do not make logical leaps in the text. Read your abstract several times and check if there is a logical connection from the beginning to the end. The abstract is supposed to draw the attention of the reader, your goal is to write an abstract that makes the reader wanting to read the entire report. Do not go too far into details; if you want to provide data, do it, but express it in a simple way (e.g., a single percentage in a sentence): do not bore the reader with data that he or she cannot understand yet. Organize the abstract into paragraphs: the paragraphs are always 3 to 5 lines long. In LaTeXsource file, go new line twice to start a new paragraph in the PDF. Do not use to go new line, just press Enter. In the PDF, there will be no gap line, but the text will go new line and a Tab will be inserted. This is the correct way to indent a paragraph, please do not change it. Do not put words in **bold** here: for emphasis, use *italic*. Do not use citations here: they are not allowed in the abstract. Footnotes and links are not allowed as well. DO NOT EVER USE ENGLISH SHORT FORMS (i.e., isn't, aren't, don't, etc.). Take a look at the following links about how to write an Abstract:

- `https://writing.wisc.edu/handbook/assignments/writing-an-abstract-for-your-research-paper/`
- `https://www.anu.edu.au/students/academic-skills/research-writing/journal-article-writing/writing-an-abstract`

Search on Google if you need more info.

CHAPTER 1

Generic Chapter

This is a generic chapter of your thesis. Remember to put ANY chapter in a different source file (including introduction and all the others).

For the purpose of this guide, the main LaTeXconstructs and how to use them will be explained here. Other thematic chapters will follow, i.e., which will trace the chapters that should be present in your thesis. Delete this generic chapter once you have learned this contents.

You can write in italic *like this*, you can write in bold **like this**, or you can write using colors like this.

This is an *itemize*, where you can put a list of items, like this:

- item number 1
- item number 2

This is an *enumerate*, where you can put a list of items with numbers, like this:

1. item number 1
2. item number 2

You can cite references like this: [1] [2], by using the \cite directive. You have to copy within \cite brackets the label of the entry that you have in the BibTeX file (`.bib`). The `.bib` file of this thesis is `mybib.bib`. he command \addbibresource at the top of this main file indicates what BibTeX file you are referring to.

As an example, this is a BibTeX entry:

```
@inproceedings{urias2018cyber,
  title={Cyber Range Infrastructure Limitations and Needs of Tomorrow: A Position Paper},
  author={Urias, Vincent E and Stout, William MS and Van Leeuwen, Brian and Lin, Han},
  booktitle={2018 International Carnahan Conference on Security Technology (ICCST)},
  pages={1--5},
  year={2018},
  organization={IEEE}
}
```

For every online paper that you may read on online libraries, you can download its BibTeX entry. For example:

1. For IEEE Xplore, click on the paper name, then click on “Cite This”, “BibTeX”, and you can find the entry;

2. For Google Scholar, click on the "Cite" voice under the paper name, then click "BibTeX", and you can find the entry.

Just copy and paste such an entry in the .bib file. If you find a paper on Scholar that is nevertheless published by IEEE, by convention you should take the entry from the IEEE website and not from Scholar. To do this, just click on the title of the paper. This will redirect you to the resource page on IEEE Xplore. Once here, follow instructions at point 1.

When you compile, a correct number will automatically be assigned to the citation in the text, and the complete entry will appear at the bottom of the document, in the "Bibliography" chapter.

If you need to cite a generic online resource, which does not necessarily correspond to a scientific paper, use the @misc entry in the `.bib` file. A @misc entry looks like this:

```
@misc{nist2018,
    author = "{NIST}",
    title = "Cyber Ranges",
    year = "2018",
    howpublished = "\url{https://www.nist.gov/system/files/documents/2018/02/13/cyber_ranges.pdf}",
    note = "[Online; Accessed 2019, 28 November]"
  }
```

You have to manually create this entry from scratch and manually type these fields. Remember not to forget any of these fields. You can choose the label with which to refer to the resource. The title of the website (which you can see at the top of the tab of your browser showing the page) can be used as the title of the resource.

In general, enter a citation of this type for sites only when there are data, phrases, or images that you intend to report. Instead, if you want to cite names of software or hardware devices, prefer the use of the \footnote, in which you will only have to specify the URL of the item.

Remember that citations, both in the text and in the image captions, usually go to the end of a sentence, before the fullstop, as in this case [**?**]. In case of long periods, they can also be placed before other detachment signs, such as commas or semicolons, or colons if they precede a list, itemized or enumerated. An exemption is allowed in the event that the name of research projects, described in some scientific resource, is being introduced, as in this case:

Cybertropolis [**?**] is described in a very good paper by Gary Deckard.

Remember to put citations very often to justify your claims, especially when you report data or results. Just consider them as a justification of what you, in an original way, are writing. Citations are not needed to have permission to copy and paste sentences from online resources, which should NEVER be done - always try to rephrase the concept with your words.

This is an image example. Images must ALWAYS be understandable: never introduce images that have text smaller than the text in your document. If you create the images yourself, try not to make them clash too much with the style of your document, and use the same font as this thesis. If they are not images of your own creation, you MUST reference them. In the caption of the image, you need to insert a citation to the resource from which you took the image, at the end of the caption sentence, before the fullstop. Each image you enter MUST be referenced in the text, using a formula similar to this:

Figure 1.1 describes the architecture of the system.

You can refer to the image using \ref followed by the image label, that you put in the \label entry of the figure. Remember to use the word Figure with a capital F.

Remember that the more your text is adorned with figures, the more understandable, appreciable and readable it becomes.

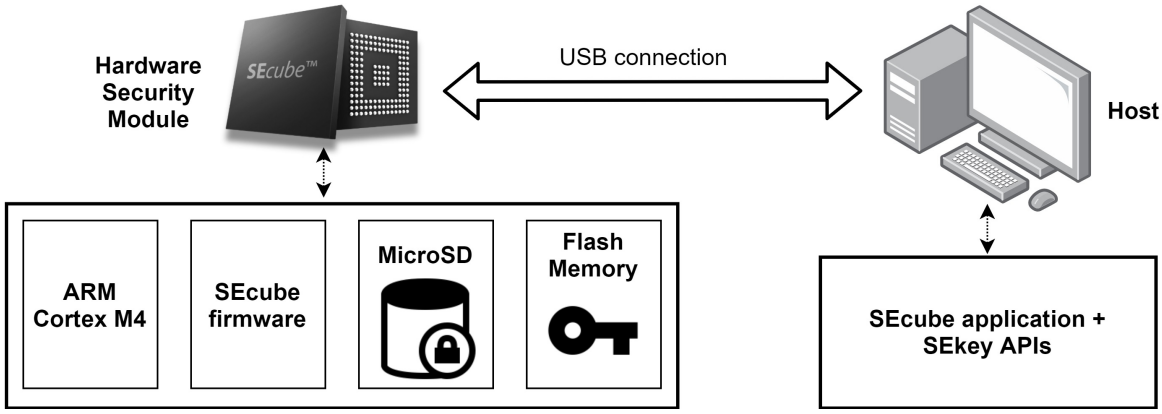

Figure 1.1: This is the image *caption*.

Table 1.1: Preliminary Experimental Results

Benchmark	**Inputs**	**Processing time**
SHA	Message of 100 KB	368449 s
RIJNDAEL	Message of 100 KB	1083568 s
DIJKSTRA	Matrix of 100x100 32-bit integers	324782 s
STRING	1331 50-char strings	178616 s
BITCOUNT	12800 32-bit integers	419545 s

1.1 Section title

This is a section under a chapter. The number of sections also contributes to greater readability of your text, and to a better display of the content in the index. In fact, sections are automatically shown in the Table of Contents. However, try not to make sections shorter than two pages. For smaller portions of your text, use subsections.

You can refer to a section using its label, using the \ref directive as for images, like this:

This concept has been explained in Section 1.1.

Remember to use the word Section with a capital S. This is also valid for chapters.

1.1.1 Subsection title

This is a subsection under the section.

The following is a table.

If you want to write a formula, you can do like this:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-ik\frac{2\pi}{N}n} \qquad k = 0, \ldots, N-1 \tag{1.1}$$

Tables and formulas are extensively documented online, and any doubts about their syntax can be easily resolved with a simple search. As for figures and sections, the same rules also apply to tables

and formulas: mandatory reference in the text, possibility to use \label to label them, and naming with capital letter (e.g., “as in Table 1.1, as in Formula 1.1).

The following is a piece of code:

```
int func(int N, int M) {
  float (*p)[N][M] = malloc(sizeof *p);
  if (!p)
    return -1;
  for (int i = 0; i < N; i++)
    for (int j = 0; j < M; j++)
      (*p)[i][j] = i + j;
  print_array(N, M, p);
  free(p);
  return 1;
}
```

You can customize the style of your code, changing the language, the colors of keywords, of comments or the background by changing the settings inside the \lstset directive found in the main file. Usually, the listings are not referenced within the text as happens for figures, tables, formulas and sections. Do not overdo the code within your text: use it only for short passages (e.g., function prototypes, or 2 to 5 lines of code within a function to help the reader in better understanding the meaning of the text).

You can also write in-text code using the \lstinline directive,
like this: `int main(int argc, char** argv);`.

CHAPTER 2

Introduction

In this first chapter we expect you to introduce the project explaining what the project is about, what is the final goal, what are the topics tackled by the project, etc.
The introduction must not include any low-level detail about the project, avoid sentences written like: we did this, then this, then this, etc.
It is strongly suggested to avoid expressions like 'We think', 'We did', etc...it is better to use impersonal expressions such as: 'It is clear that', 'It is possible that', '... something ... has been implemented/-analyzed/etc.' (instead of 'we did, we implemented, we analyzed').
In the introduction you should give to the reader enough information to understand what is going to be explained in the remainder of the report (basically, expanding some concept you mentioned in the Abstract) without giving away too many information that would make the introduction too long and boring.
Feel free to organize the introduction in multiple sections and subsections, depending on how much content you want to put into this chapter.

Remember that the introduction is needed to make the reader understand what kind of reading he or she will encounter. Be fluent and try not to confuse him or her. The introduction must ALWAYS end with the following formula: The remainder of the document is organized as follows. In Chapter 2, ...; in Chapter 3, ... so that the reader can choose which chapters are worth skipping according to the type of reading he or she has chosen.

CHAPTER 3

Background

In this era where digital security is very important, hardware-based security is on the rise because provide very complex and reliable solutions. Physically Unclonable Functions (PUFs) have emerged as a promising approach to address the challenges of hardware authentication, secure key generation, and anti-counterfeiting measures. Among the various PUF implementations, Complementary Metal-Oxide-Semiconductor Arbiter PUFs (CAMPUFs) stand out as a powerful hardware security primitive based on CMOS technology.

CAMPUFs leverage the unique physical variations inherent in CMOS integrated circuits to generate unpredictable and practically unclonable responses. The foundation of CMOS technology, with its low power consumption and high integration capabilities, makes it an ideal platform for building complex digital circuits and implementing PUFs.

This documentation will present and underly principles of PUFs, explain the workings of CAMPUFs.

3.1 General Concepts

- **PUF**: Physically Unclonable Function (PUF) is a security metric that exploits inherent device physical variations to produce an unclonable, unique device response to a given input. Unclonability means that each PUF has a unpredictable response to stimuli, because the response is created by complex interactions between many random components. PUFs implement a challenge-response type of authentication. They act as a unique device identifier. //(TO WRITE: EXAMPLES OF PUFs)

- **Fixed Pattern Noise**: it is a particular noise pattern present on digital imaging sensors, caused by small differences in specific sensor pixels, patterns can result in brighter or darker pixels. It is created from two sources: PRNU (Photo Response Non-Uniformity) and DSNU (Dark Signal Non-Uniformity). The latter will be the focus of our PUF implementation.

- **PRNU**:

- **DSNU**: Dark Signal Non-Uniformity is a type of noise that occurs in dark images. The intensity values for pixels does not start from zero, an offset is added on every pixel, to avoid signal values dropping lower than zero. This makes each pixel to always have a non-zero value, which is the *bias*. Fluctuation in the bias is the DSNU

- **Challenge-Response Authentication**: authentication method based on a unique challenge provided to a device by an authenticator. The device must create a response to the challenge,

based on unique physical properties that only that specific device has.

•

Physically unclonable functions (PUFs) are a technique in hardware security that exploits inherent device variations to produce an unclonable, unique device response to a given input.

3.1.1 Purposes

Key Generation: PUFs can generate cryptographic keys directly from the unique physical responses of the ICs. These keys can be used for secure communication, encryption, and data protection.

Challenge-Response Mechanism: PUFs operate on a challenge-response mechanism, where a unique challenge is provided to the device, and the corresponding response is generated based on the unique physical characteristics of that particular IC.

Randomness and Entropy Generation: PUFs can serve as a high-quality source of randomness and entropy, critical for cryptographic protocols and secure communication.

Authentication: PUF responses can be used for secure device authentication, verifying the identity and integrity of hardware components in various applications, such as secure booting and secure firmware updates.

Uniqueness and Unpredictability: PUFs generate device-specific responses based on the inherent physical variations during manufacturing. As a result, each instance of the IC exhibits a unique response, making it practically impossible to clone or replicate the device.

Anti-Counterfeiting Measures: PUFs play a crucial role in preventing counterfeiting and unauthorized duplication of hardware components, as the uniqueness of the responses ensures the authenticity of genuine devices.

3.1.2 Security Properties and Applications of PUFs

Key Security Properties of PUFs

Unpredictability: The inherent randomness linked to PUF responses, even when given the same challenge multiple times. This property makes stonger the security of PUF-based authentication and key generation.

Uniqueness and Unclonability: PUFs depend on the uniqueness of their physical microstructure. This microstructure depends on random physical factors introduced during manufacturing that gives unique reponses. These factors are unpredictable and uncontrollable, which makes it virtually impossible to duplicate or clone the structure.

Resistance to Physical Attacks: PUFs are resilient against various physical attacks, including invasive attacks like reverse engineering and probing, as well as non-invasive attacks like side-channel analysis.

Practical Applications of PUFs

Secure Key Generation: PUF responses can be utilized to generate cryptographic keys without the need for additional storage of secret keys, enhancing the security of cryptographic protocols.

Hardware Authentication: PUFs can be employed for secure device authentication, ensuring the legitimacy of hardware components and protecting against unauthorized access.

Anti-Counterfeiting Measures: PUFs serve as a powerful tool for detecting counterfeit devices, as the unique responses enable the verification of genuine products.

Secure Communication: PUF-based keys are valuable for securing communication channels, enabling secure and authenticated data transmission.

3.1.3 Categories of Implementation

PUFs can be categorized based on different operational principles and sources of randomness

Intrinsic PUFs

Intrinsic PUFs are characterized by their reliance on the inherent physical variations present within a single chip during the semiconductor manufacturing process. These variations arise due to manufacturing imperfections, process fluctuations, and random dopant fluctuations. The unique characteristics of each individual chip create a fingerprint-like response, making Intrinsic PUFs ideal for hardware authentication and identification purposes.

Common Implementations of Intrinsic PUFs are:

- Delay PUFs: Delay PUFs exploit the differences in signal propagation delay along various paths within the circuit. By measuring these delays, a set of unique challenge-response pairs can be generated, forming the basis for secure authentication.
- Ring Oscillator PUFs: Ring oscillator PUFs use the frequency differences in ring oscillators, circuits comprising an odd number of inverters. The varying delays in these oscillators result in distinctive response patterns, enabling the generation of cryptographic keys and unique device identifiers.
- SRAM PUFs: SRAM PUFs exploit the randomness in the power-up behavior of standard static random-access memory on a chip as a PUF.
- VIA PUFs: the Via PUFs technologies are based on "via" or "contact" formation during the standard CMOS fabrication process.

Extrinsic PUFs

Extrinsic PUFs differ from Intrinsic PUFs as they derive their responses from external environmental factors that influence the behavior of the chip. These external factors may include temperature variations, light levels, power supply fluctuations, and electromagnetic interference. The responses generated by Extrinsic PUFs can change under different operating conditions, leading to additional sources of randomness.

Common Implementations of Extrinsic PUFs are:

- Ambient Light PUFs: Ambient light PUFs utilize the incident light level on the chip's surface to create distinct response patterns. Variations in light intensity cause corresponding variations in the generated responses, enabling unique identification.
- Temperature PUFs: Temperature PUFs exploit temperature-induced changes in the electrical characteristics of the chip. Different temperature levels result in varied responses, contributing to the unclonable behavior of the device.

3.2 Technical Manual

(Aimed at engineers, technical users that work directly with source code etc.)

(Provide architectural overview of the project components and explain their interactions)

(code structure, key modules + tools required to build project and run it)

This camPUF implementation is based on CMOS image sensors, typically present on modern smartphone cameras. The entirety of the project consists of a digital signal processing part and a device authentication part. All the project is implemented using Python3 Here is the list of the libraries used:

- Opencv: computer vision library, used to extract data from raw images

- scipy: python library that provides mathematical instruments and algorithms
- MatPlotlib: graph library for math visualizations

There are two main python modules, one for DSNU extraction and one for the enrollment procedure.

CHAPTER 4

Implementation Overview

In this chapter you should provide a general overview of the project, explaining what you have implemented staying at a high-level of abstraction, without going too much into the details. Leave details for the implementation chapter. This chapter can be organized in sections, such as goal of the project, issues to be solved, solution overview, etc.
It is very important to add images, schemes, graphs to explain the original problem and your solution. Pictures are extremely useful to understand complex ideas that might need an entire page to be explained.
Use multiple sections to explain the starting point of your project, the last section is going to be the high-level view of your solution...so take the reader in a short 'journey' to showcase your work.

CHAPTER 5

Implementation Details

This is where you explain what you have implemented and how you have implemented it. Place here all the details that you consider important, organize the chapter in sections and subsections to explain the development and your workflow.
Given the self-explicative title of the chapter, readers usually skip it. This is ok, because this entire chapter is simply meant to describe the details of your work so that people that are very interested (such as people who have to evaluate your work or people who have to build something more complex starting from what you did) can fully understand what you developed or implemented.
Don't worry about placing too many details in this chapter, the only essential thing is that you keep everything tidy, without mixing too much information (so make use of sections, subsections, lists, etc.). As usual, pictures are helpful.

CHAPTER 6

Results

In this chapter we expect you to list and explain all the results that you have achieved. Pictures can be useful to explain the results. Think about this chapter as something similar to the demo of the oral presentation. You can also include pictures about use-cases (you can also decide to add use cases to the high level overview chapter).

6.1 Known Issues

If there is any known issue, limitation, error, problem, etc...explain it in this section. Use a specific subsection for each known issue. Issues can be related to many things, including design issues.

6.2 Future Work

Adding a section about how to improve the project is not mandatory but it is useful to show that you actually understood the topics of the project and have ideas for improvements.

CHAPTER 7

Conclusions

This final chapter is used to recap what you did in the project. No detail, just a high-level summary of your project (1 page or a bit less is usually enough, but it depends on the specific project).

Bibliography

[1] Donald E. Knuth (1986) *The TEX Book*, Addison-Wesley Professional.

[2] Leslie Lamport (1994) *LATEX: a document preparation system*, Addison Wesley, Massachusetts, 2nd ed.

APPENDIX A

User Manual

In the user manual you should explain, step-by-step, how to reproduce the demo that you showed in the oral presentation or the results you mentioned in the previous chapters.
If it is necessary to install some toolchain that is already well described in the original documentation (i.e., Espressif's toolchain for ESP32 boards or the SEcube toolchain) just insert a reference to the original documentation (and remember to clearly specify which version of the original documentation must be used). There is no need to copy and paste step-by-step guides that are already well-written and available.
The user manual must explain how to re-create what you did in the project, no matter if it is low-level code (i..e VHDL on SEcube's FPGA), high-level code (i.e., a GUI) or something more heterogeneous (i.e. a bunch of ESP32 or Raspberry Pi communicating among them and interacting with other devices).

APPENDIX B

API

If you developed some source code that is supposed to be used by other software in order to perform some action, it is very likely that you have implemented an API. Use this appendix to describe each function of the API (prototype, parameters, returned values, purpose of the function, etc).