Politecnico di Torino

Cybersecurity for Embedded Systems
01UDNOV

Master's Degree in Computer Engineering

Design and Development of a RAM-based PUF

Project Report

Candidates:
Zissis Tabouras (s284685)
Elena Roncolino (s304719)
Stefano Palmieri (s281677)

Referents:
Prof. Paolo Prinetto
Dr. Matteo Fornero
Dr. Vahid Eftekhari

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

Introduction

In the last years, the number of small electronic devices that can be connected with big computational units grew exponentially. Embedded systems play a crucial role in fueling the growth of the Internet-of-Things (Iot) in the most diverse domains, such as health care, home automation and transportation. By the end of 2022 the number of IoT devices connected to the Internet is expected to reach the astonishing number of 14.4 billions [1]. The ubiquitousness of such devices coupled with their ability to access potentially sensitive/confidential information has given rise to security and privacy concerns. An additional challenge is the growing number of counterfeit components in these devices, resulting in serious reliability and financial implications.

Physical unclonable functions (PUFs) are a promising security primitive to help address these concerns. PUFs extract secrets from physical characteristics of integrated circuits (ICs) [2] and therefore require minimal or no additional hardware for their operation and are therefore cheaper than other solutions. The instance-specific nature of the secret provide a mean to to uniquely identify and authenticate each device based on a challenge-response mechanism [3].

The aim of this project is to design and develop a RAM based PUF for the SEcube$^{\text{TM}}$, a single-chip easily integratable device capable of hiding significant complexity behind a set of simple high-level APIs [4].

The remainder of the document is organized as follows:

In Chapter 2, a brief background and state of the art of this topic is provided;
In Chapter 3, an implementation overview is presented;
In Chapter 4, implementation details are described;
In Chapter 5, results are listed;
In Chapter 6, conclusions and final observations are presented.
Appendix A describes how a demo of the implementation can be run.
Appendix B describes the APIs created for this project.

CHAPTER 2

Background

2.1 State of the art of embedded systems security approach

The current best practice for providing a secure memory or authentication source in mobile systems is to place a secret key in nonvolatile electrically erasable programmable read-only memory (EEPROM) or battery-backed static random-access memory (SRAM) and use hardware cryptographic operations such as digital signature or encryption. Nonetheless, this approach is expensive both in terms of design and of power consumption. In addition, invasive attack mechanisms make such nonvolatile memory vulnerable. Protection agains such attacks is therefore needed and it requires the use of active tamper detection/prevention circuitry which must be continually powered [2].

2.2 Physical unclonable functions

Physical unclonable functions (PUFs) are innovative primitives that derive secrets from complex physical characterstics of the ICs rather that storing the secret in digital memory. Because the PUF taps into the random variation during an IC fabrication process, the secret is extremely difficult to predict or extract. PUFs generate volatile secrets that only exist in a digital form when a chip is powered on and running. This requires the adversary to mount the attack while the IC is running and using the secret, which is significantly harder that discovering non-volatile keys. An invasine attack must measure the PUF delays without altering them or triggering sensing wires that clear out the registers [5].

The concept of PUFs is based on the idea that even though the mask and manufacturing process is the same during the creation of the same type of IC, each IC is actually slightly different due to normal manufacturing variability. PUFs leverage this variability to derive the silicon ”biometric”, a ”secret” information that is unique to the chip. This implies that no two identical chips can be manufactured. Although the use of PUFs is a relatively new technology, it should be noted that the concepts of unclonability and uniqueness of ojects have been extensively used in the past [2].

2.2.1 Types of PUFs

Most of the currently used PUFs fall into two categories:

- strong PUFs, mainly used for authentication, and
- weak PUFs, primarily used for key storage.

A PUF can be modeled as a black-box challenge-response system: an input challenge c is passed to a PUF which returns a response $r = f(c)$, where $f(\cdot)$ describes the input/output relations of the PUF. The black-box model is appropriate to describe PUFs since input parameters of $f(\cdot)$ are hidden from the user since they represent the interfan manufacturing variability that PUFs use to generate unique challenge-response sets.

The fundamental difference between weak and strong PUFs is the domain of $f(\cdot)$, i.e., the number of unique challenges c that the PUF can process. Weak PUFs can only support a small number of challenges (in some cases just a single challenge). On the contrary, a strong PUF can support a large enough number of challenges such that trying to determine/measure all challenge/response pairs (CRPs) within a limited timeframe is unfeasible.

Both weak and strong PUFs rely on analog physical properties of the fabricated circuit to derive secret information and therefore have noise and variability associated with them. For this reason, modern PUFs designs employ multiple error-correction techniques to mitigate the noise and improve realiability.

Examples of strong PUFs include optical and arbiter PUFs, while ring-oscillator and SRAM PUFs are example of weak ones. [2]

2.2.2 SRAM PUF

SRAM PUFs exploit the positive feedback loop in a SRAM. A SRAM has two stable states (used to store a 1 or a 0), and positive feedback to force the cell into one of these two state and to prevent an accidental state transition.

Figure 2.1 shows a common six-transistor configuration of an SRAM consisting of cross-coupled CMOS inverters (M_1-M_4) and access transistors (M_5-M_6).

Theoretically, when a device with a SRAM is powered on and no write operation is performed, the SRAM cell exists in a metastable state where the feedback pushing the cell toward the "1" state equals the feedback pushing the cell toward the "0" state, thereby keeping the cell indefinitely in this metastable state. However, in actual implementations one feedback loop is always slightly stronger than the other due to small transistor threshold mismatches resulting from process variation. This means that the cell at start up relaxes into either the "1" or "0" state. The final state of the cell depends on the difference between two feeback loops and it is therefore not strongly impacted by temperature or power supply fluctuations. Nonetheless, if the two feedback loops are sufficiently close then random noise or other small environmental fluctuations can result in an output bit flip. Therefore, error correction of this output will be necessary. Error correction can be performed by using repeated measurement: since the relative strengths between the two feedback is relatively static, by measuring the outputs of the cell repeatedly one can assess the stability of a SRAM PUF bit and selectively use the most stable bits as the PUF output. [2]

2.3 SEcube^{TM}

The SEcube^{TM} (Secure Environment cube) Open Security Platform is an open source security-oriented hardware and software platform. It provides hardware and software holistic security focusing on common operational security concepts like groups and policies instead of classical security concepts such as cryptographic algorithms and keys [6]. The SEcube^{TM} is the smallest reconfigurable silicon that combines three main cores in a single-chip design. It embeds a low-power ARM Cortex-M4 processor, a flexible and fast Field-Programmable-Gate-Array (FPGA), and an EAL5+ certified Security Controller (SmartCard), as shown in Figure 2.2. This make the SEcube a secure environment since it is based on a modular software architecture where all functions are isolated [7].

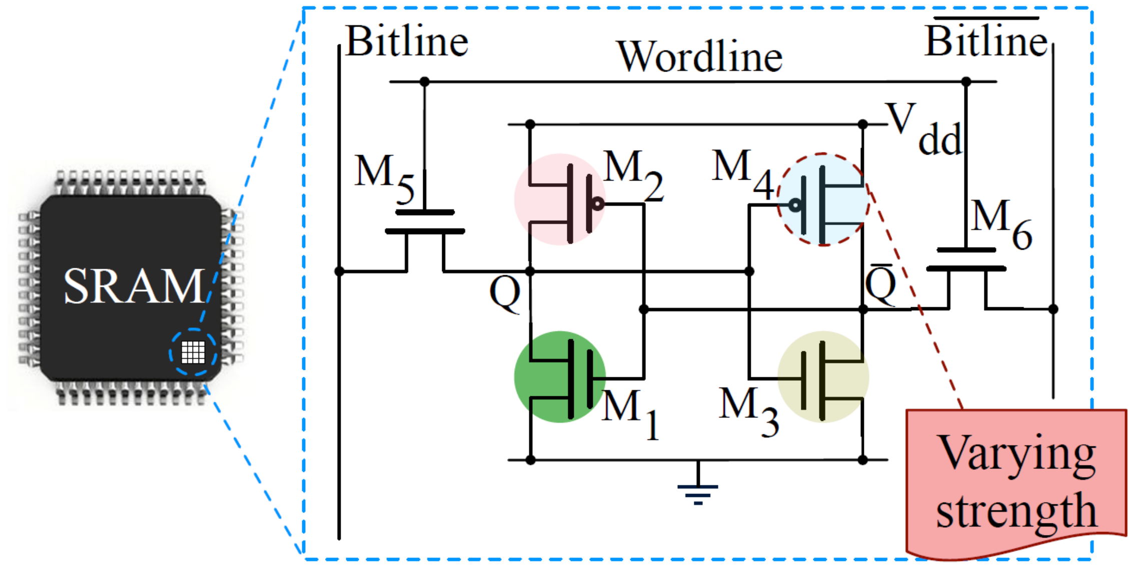

Figure 2.1: SRAM bit cells [3].

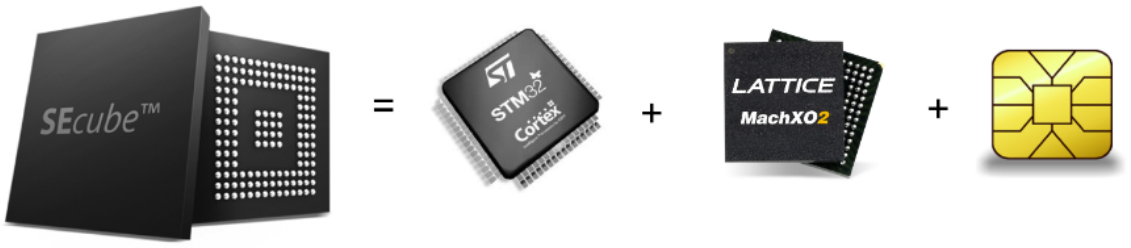

Figure 2.2: The three components of the SEcube^{TM}: the ARM Cortex-M4 processor, the FPGA and the EAL5+ SmartCard[6].

CHAPTER 3

Implementation Overview

As already said the aim of the project is to provide a secure PUF to recognize IoT devices, in order to avoid impersonation attacks.

The type of PUF implemented is a SRAM PUF (parlare un attimo dell' SRAM PUF se non e' stato fatto prima), this was implemented using a SECube device.

The Idea is that the first time an Host is connected to the SECube, it asks the device all the PUFs that it has collected during its starting phases. This challenge and response information has to be stored by the host in a cipher way:

$$data\_to\_store = H(response) \tag{3.1}$$

the challenge has to stay in plaintext.

In the future when the host has to establish a connection with the device, he is going to send to it a specific challenge, the device is going to answer with a response; then the host has to check the validity of the response, evaluating the digest and comparing with the one that he has in the storage file. Once a challenge-response is used, it has to be eliminated and not used anymore in the future; in this way it is possible to avoid replicant attacks.

The implementation of this project can be divided in two flow:

1. The first flow consists in the exchange of all the challenge and response information between host and device

2. The second flow consists in the challenge and response authentication of the device

Here will be explained only a general idea of the implementation and in the next chapter will be described deeply the functioning.

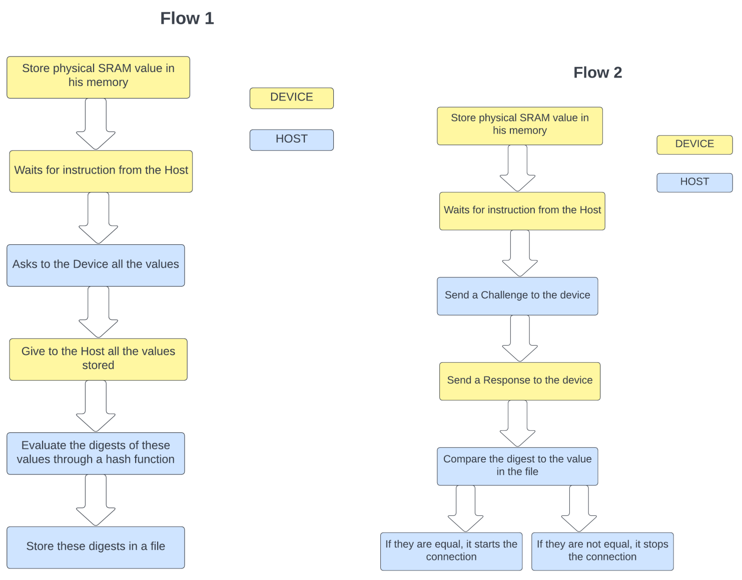
Flow 1
Store physical SRAM value in his memory
Waits for instruction from the Host
Asks to the Device all the values
Give to the Host all the values stored
Evaluate the digests of these values through a hash function
Store these digests in a file
DEVICE
HOST
Flow 2
Store physical SRAM value in his memory
Waits for instruction from the Host
Send a Challenge to the device
Send a Response to the device
Compare the digest to the value in the file
If they are equal, it starts the connection
If they are not equal, it stops the connection
DEVICE
HOST

Figure 3.1: Flow 1 and Flow 2

3.1 Device side

On the device side both of the flows have a common important operation that has to be done. This operation consists in taking the values present in the SRAM when this one is switched on and storing them in a secure place. Obviously this operation is the first operation that has always been done by the device as soon as it is switched on.

Secondly, if this is the first time that it communicates with this particular host it waits until the host asks it for all the value that it has just taken from the SRAM in order to implement the challenge-response authentication for the next time.

On the other hand, it waits until the host sends to it a particular challenge, this challenge will be the index of a determined response, so the device takes the response and sends them to the host. After that the device is authenticated and the can starts to communicate.

3.2 Host side

On the host side the type of work is a little bit different from the device side. When the connection is instantiated with the device for the first time it asks the device all the values needed to implement the challenge-response authentication; the Host is going to store the hash values of these datas in a file.

During all the next connections with the device, the Host sends a particular challenge to the device and waits for the respective response. After it receives it, It is going to evaluate the digest of this value, it gives it in input to the hash function and it compares it with the value that it has inside the file.

If the two digests are the same it means that the device is the correct one and so the connection can start; otherwise the host stops the connection.

CHAPTER 4

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 5

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).

5.1 Known Issues

One many issue of this kind of approach could be that there is not the possibility to avoid a Man-in-the-Middle, in order to avoid that a man can steal information from this kind of information it is necessary to encrypt the communication. It is important to say that the type of encryption and the necessity to encrypt or not depend from the type of device and by the level of sensibility of the datas.

5.2 Future Work

Many are the implementations that can be done on this project. The main one is to evaluate and store in a secure place the hash value of the file containing the challenge-response. This kind of implementation can be used in order to ensure the integrity of the challenge-response of a particular device. The idea consists in evaluating the hash value of the file before taking information from it and comparing it with the digest that we store in another place. If the value is the same it means that the file is not corrupted.

CHAPTER 6

Conclusions

To conclude, with this project it has been shown how a sram PUF works and how to implement it. In particular at the beginning It has been presented the problem present in these years, and why this kind of solution is important.

Subsequently has been shown some different kinds of PUF and how to approach the implementation of one of them....

It has been shown how to increase the security of the implementation with some simple precautions as a hash function that can help to increase integrity... .....

In the end it was shown some results of this approach....

To continue——————————————-

CHAPTER 7

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: [?] [?], 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 7.1 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.

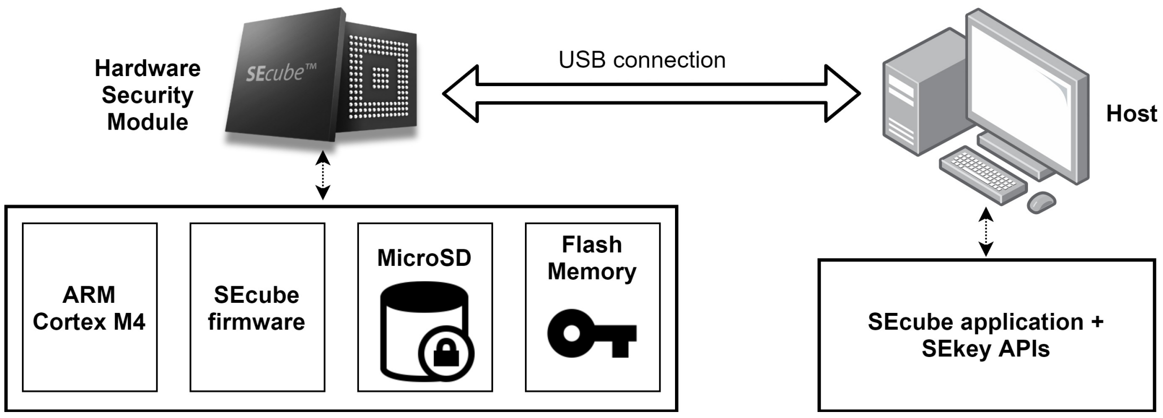

Figure 7.1: This is the image *caption*.

Table 7.1: 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

7.1 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 7.1.

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

7.1.1 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:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-ik\frac{2\pi}{N}n} \qquad k = 0, \dots, N-1 \tag{7.1}$$

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 7.1, as in Formula 7.1).

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);`.

Bibliography

[1] IoT.Business.News, *State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally*, 2022, `https://iotbusinessnews.com/2022/05/19/70343-state-of-iot-2022-number-of-connected-iot-devices-growing-18-to-14-4-billion-globally/` [Online; Accessed 2022, 29 July].

[2] C. Herder, M. Yu, F. Koushanfar and S. Devadas, *Physical Unclonable Functions and Applications: A Tutorial*, in *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126-1141, Aug. 2014, doi: 10.1109/JPROC.2014.2320516.

[3] S. Sutar, A. Raha, and V. Raghunathan, *Memory-based Combination PUFs for Device Authentication in Embedded Systems*, for the School of Electrical and Computer Engineering, Purdue University, 5 December 2017, arXiv:1712.01611v1

[4] A. Varriale, E. I. Vatajelu, G. Di Natale, P. Prinetto, P. Trotta and T. Margaria, *SEcube™: An open-source security platform in a single SoC*, 2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS), 2016, pp. 1-6, doi: 10.1109/DTIS.2016.7483810.

[5] G. E. Suh and S. Devadas, *Physical Unclonable Functions for Device Authentication and Secret Key Generation*, 2007 44th ACM/IEEE Design Automation Conference, 2007, pp. 9-14.

[6] M. Fornero, N. Maunero, P. Printetto, G. Roascio, A. Varriale, *SEcube^{TM} Open Security Platform, Introduction.* Released October 2021. `https://github.com/SEcube-Project/SEcube-SDK`.

[7] *What is SEcube^{TM}*, `https://www.secube.blu5group.com/`. [Online; Accessed 2022, 05 August].

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

[9] 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).