UNIVERSIDADE D
COIMBRA

Pedro Martins

Encryption algorithms for resource-constrained devices

Dissertation in the context of the Master in Informatics Security, advised by Professor Vasco Pereira, Professor Bruno Sousa, and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July 2024

DEPARTAMENTO DE
ENGENHARIA INFORMÁTICA
FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Pedro Martins

ENCRYPTION ALGORITHMS FOR RESOURCE-CONSTRAINED DEVICES

Dissertation in the context of the Master in Informatics Security, advised by Professor Vasco Pereira, Professor Bruno Sousa, and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July 2024

DEPARTAMENTO DE
ENGENHARIA INFORMÁTICA
FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Pedro Martins

ALGORITMOS DE ENCRIPTAÇÃO PARA DISPOSITIVOS COM RECURSOS LIMITADOS

Dissertação no âmbito do Mestrado em Segurança Informática, orientada pelo Professor Vasco Pereira e Professor Bruno Sousa e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho 2024

Funding

This work has been supported by Project “Agenda Mobilizadora Sines Nexus”. ref. No. 7113), supported by the Recovery and Resilience Plan (PRR) and by the European Funds Next Generation EU, following Notice No. 02/C05-i01/2022, Component 5 - Capitalization and Business Innovation - Mobilizing Agendas for Business Innovation.

Acknowledgements

I want to express my sincere gratitude to all the people who have contributed to completing this thesis. Their support, guidance, and encouragement have been invaluable. First and foremost, I would like to thank my advisors, Professor Vasco Pereira, Professor Bruno Sousa, and Khan Saad for all their help and guidance throughout the entire year, which were crucial for the development of the thesis.

Special thanks for all the help and support provided at Departamento de Engenharia Informática and its many members and colleagues who guided and supported me not only with the development of this thesis but also with my academic studies in this area.

I would also like to leave a special word for my friends and family who although were not directly involved in the development of this project, provided me with unwavering support throughout this new challenge.

Abstract

The Internet Of Things (IoT) is spreading rapidly across multiple sectors, bringing efficiency and energy savings to businesses and revolutionizing the way we communicate with low-cost computing. One very important use case for this technology is seaports, optimizing their operations and simultaneously reducing their overall costs. As with most devices, these systems need security to protect transmitted data from tampering and disclosure so this thesis will focus on evaluating lightweight encryption solutions, measuring and comparing both their performance and security to define specific guidelines for their usage in resource-constrained devices, with some contributions to the seaport environment. This work also gives a brief background on topics like encryption, block ciphers, and different specifications of lightweight algorithms, providing several low-resource devices, namely Arduino Uno, several Raspberry Pi devices, and ESP-family hardware, which are widely used in these IoT scenarios to assess their performance.
Furthermore, it also analyzes three transmission protocols that are used in these environments to transfer data between these devices, namely Wi-Fi, Bluetooth, and ESP-Now. To complement this, a literature review was conducted to provide context into the current state-of-the-art on the topics of seaport and encryption usage, as well as to define the methodology and subsequent metrics on which we can perform the comparisons and thus objectively propose the best solutions. Afterward, a practical assessment was performed following the methodology chosen, which determined that TEA, XTEA, and ASCON are, on average, the most efficient ciphers when it comes to speed, memory, and power consumption, with AES-128 also working as a viable solution given its stature as a highly used and standardized algorithm with decent performance. Furthermore, from the transmission standards gathered, we can conclude that Bluetooth Low Energy (BLE) is by far the most power efficient, however, it has a limited range that hinders its usability for any long-range communications, with ESP-Now and Wi-Fi offering competitive solutions if power saving is not critical and a medium to long range is desirable. To conclude, scenarios that fit both into the IoT and seaport use cases were defined to suggest the best possible solutions for different situations, most originating from the state-of-the-art analysis.

Keywords

Lightweight encryption, Arduino, Raspberry Pi, Seaport, Internet of Things

Resumo

A Internet das Coisas (IoT) está a espalhar-se rapidamente por vários setores, trazendo eficiência e poupança de energia às empresas e revolucionando a forma como comunicamos a utilizar computação de baixo custo. Um caso de uso muito importante para esta tecnologia são os portos marítimos, otimizando as suas operações e reduzindo simultaneamente os seus custos globais. Tal como acontece com a maioria dos dispositivos, estes sistemas necessitam de segurança para proteger os dados transmitidos contra adulteração e divulgação, pelo que esta tese se concentrará na avaliação de soluções de encriptação leves, medindo e comparando o seu desempenho e segurança para definir diretrizes específicas para o seu uso em dispositivos com recursos limitados, com algumas contribuições para as operações de portos marítimos. Este trabalho também fornece um breve histórico sobre tópicos como encriptação, cifras de bloco, diferentes especificações de algoritmos leves, assim como utilizar vários dispositivos com poucos recursos, nomeadamente Arduino, vários Raspberry Pi, e dois dispositivos ESP que são amplamente utilizados nestes cenários de Internet das Coisas para avaliar o seu desempenho. Além disso, também serão estudados três protocolos de transmissão que são utilizados nestes ambientes para transferir os dados entre estes dispositivos, nomeadamente Wi-Fi, Bluetooth e ESP-Now. Para complementar isso, foi feita uma revisão da literatura para contextualizar o estado da arte atual sobre os tópicos de porto marítimo e uso de criptografia, bem como para definir a metodologia e métricas subsequentes nas quais nós podemos realizar as comparações e assim propor objetivamente as melhores soluções. Posteriormente, foi realizada uma avaliação prática seguindo a metodologia escolhida, que determinou que TEA, XTEA e ASCON são, em média, as cifras mais eficientes em termos de velocidade, memória e consumo de energia, com o AES-128 funcionando também como uma solução viável dada a sua estatura como um algoritmo altamente utilizado e padronizado na indústria, aliado a um desempenho aceitável. Além disso, a partir dos padrões de transmissão recolhidos, podemos concluir que o BLE é de longe o mais eficiente em termos energéticos, no entanto, tem um alcance limitado que dificulta a sua utilização para quaisquer comunicações de longo alcance, com o ESP-Now e Wi- Fi oferecendo soluções competitivas se a poupança de energia não for crítica e um alcance médio a longo for desejável. Para concluir, foram definidos cenários que se enquadram tanto nos casos de uso de IoT quanto de portos marítimos para sugerir as melhores soluções possíveis para diferentes situações, a maioria com origem na análise do estado da arte.

Palavras-Chave

Encriptação leve, Arduino, Raspberry Pi, Portos Marítimos, Internet das Coisas

Contents

Acronyms

ABE Attribute-Based Encryption.

ARP Address Resolution Protocol.

BLE Bluetooth Low Energy.

CAESAR Competition for Authenticated Encryption: Security, Applicability, and Robustness.

CFB Cipher Feedback.

CPU Central Processing Unit.

DSM Diffusion Switching Mechanism.

ECB Electronic Codebook.

FHSS Frequency Hopping Spread Spectrum.

FN Feistel Network.

GCM Galois/Counter Mode.

GE Gate Equivalence.

HTTP Hypertext Transfer Protocol.

IDE Integrated Development Environment.

IoT Internet Of Things.

IV Initialization Vector.

LPWAN Low-Power Wide-Area Networks.

MiTM Man in the Middle.

MQTT Message Queuing Telemetry Transport.

MSE Mean Squared Error.

MWh Megawatt Hour.

NIST National Institute of Standards and Technology.

NSA National Security Agency.

RMSE Root Mean Squared Error.

RSSI Received Signal Strength Indicator.

SAC Strict Avalanche Criterion.

SF Spreading Factor.

SHM Structural Health Monitoring.

SIG Special Interest Group.

SP Sponge Construction.

SPN Substitution Permutation Network.

SRAM Static Random Access Memory.

VHF Very High Frequency.

WAP Wireless Access Point.

List of Figures

List of Tables

Chapter 1

Introduction

The main objective of this thesis is the study of lightweight encryption algorithms for devices with limited resources, more specifically in the context of seaports and their use of IoT technology to automate and/or improve their management. These algorithms are considered lightweight given the fact that they are specially developed to use the least number of resources whilst still yielding good security and efficiency. They use symmetric encryption and consist of block ciphers. The purpose of the thesis is to select the most appropriate ones based on their security, available implementations, and overall efficiency to conduct a thorough study with practical integration on several IoT devices, namely Arduino Uno, Raspberry Pi Pico, Zero, and 3B+, and both ESP32 and ESP8266, to assess their performance and usability in scenarios that can emulate real-world applications in the seaport use case. Furthermore, three transmission standards will also be subject to testing, namely Wi-Fi, Bluetooth, and ESP-Now, in order to establish their usability within a resource-constrained environment according to power and range metrics.

The final goal is to contribute to any area that needs encryption to be integrated into scenarios with resource-constrained devices, with a thorough and documented work cycle and evaluation of available encryption algorithms, compared using several metrics to assess both their performance and security.

1.1 Motivation

With the modern surge in Internet usage and a consistent trend toward smaller and more efficient devices, a lot of research is needed to keep up with the current demand. IoT technology is on the rise, with practical applications ranging from personal to commercial and industrial use. But, with the ever-expanding domain of this technology, threats also appear that want to take advantage of this growth to inflict damage or gain access to unauthorized information. There is a current need for works that align with the profile of this type of device and can offer solutions that offer robust security mechanisms but, at the same time, do not compromise their efficiency and support fast data exchanges.

Although there are already some works documenting similar approaches to fill the lack of knowledge on this topic, they still leave some information gaps needed to assess the full scalability of the theme. Most of them either describe or compare algorithmic solutions far too heavy to be properly integrated into resource-constrained devices, lack depth in their lightweight encryption research (i.e., too few metrics are used to determine which algorithm is the most adequate), or do not have a written application on the seaport scenario and describe a more general use.

1.2 Contributions

As part of the thesis, we intend to extend the knowledge on several fronts, namely on the recommendation for encryption algorithms to be used for IoT devices that are considered resource-constrained. Furthermore, the analysis will also include tests with different transmission protocols, which will be extremely useful in assessing the best mechanisms for data transmission within the IoT context. Lastly, with the appearance of newer lightweight ciphers to answer the increasing demand for IoT technology driven by resource saving operations, this study is also going to choose a newer algorithm to compare with older ones and try to understand if there are any new solutions used that can contribute to a higher degree of performance, security, or resource-saving that are not algorithm-specific.

These contributions will also include a contribution to the scientific community of an article with a summarized version of the analysis conducted to reach the previously mentioned objectives, as well as the overall findings and potential improvements for these scenarios, with the final goal of publishing it in the IEEE Internet of Things Journal [IEEE Internet of Things Journal]. Furthermore, as part of the thesis, concise guidelines on the usage of encryption and data transmission for the use case and limited devices will be presented, to complement the knowledge gaps that exist at the moment in this regard. Thus, the objective is to produce practical recommendations supported by all the data gathered and analyzed throughout the document.

This document will also contribute to the NEXUS WP7 in the connection between encryption solutions and the seaport environment with IoT technology.

1.3 Structure

This thesis is structured into 6 different chapters that are organized in the following way:

- **Chapter 2** will briefly introduce the general concepts that will be approached in the document, known as Background. Details about the technical aspects of Symmetric Encryption using Block Ciphers and some of the more prominent lightweight algorithms that use this structure. Furthermore, some con-

text will also be provided in the profile of the IoT devices and transmission standards used to send and receive encrypted data, as well as the potential problems that may arise in the lightweight environment with quantum technology.

- **Chapter 3** consists of the State of Art and subsequent Literature Review on the topic of lightweight encryption and is subdivided into the relevant works identified that have a similar approach to the thesis, the application of IoT devices in a seaport context, and the study of USB Power Meters as a power-measuring tool with resource-constrained devices. Furthermore, it also provides the reader with the research strategy used to perform such analysis.

- **Chapter 4** includes further research into the seaport environment, with a special focus on security concerns and eliciting explicit requirements into the functionality of a seaport and its several services.

- **Chapter 5** introduces the methodology and the plan of action to conduct the evaluation and accomplish the target of the thesis. This chapter contains the implementations available for the IoT devices, the study of the algorithms' performance and security to determine which ones are better suited for the comparison process, the metrics needed to obtain quantifiable results for the cipher's performance whilst running in the devices, the three analyzed ways of transmitting data on resource-constrained devices, and finally, a concise plan of action that will lay out the future chapters and how the testing and analysis phases should be conducted, based on the knowledge acquired in previous chapters, as well as expressing the amount of time spent on both the intermediate and final versions of this thesis.

- **Chapter 6** contains the first of two parts regarding the practical assessment at hand. This chapter focuses solely on lightweight encryption and decryption for the proposed IoT devices, assessing each cipher's performance based on a multitude of criteria established in the previous chapter and producing a critical analysis of their time and power consumed on each hardware. Furthermore, each device's performance is also analyzed to confirm its correlation with the technical specifications provided. Lastly, some scenarios involving technology used in seaports and discussed in the state-of-the-art are presented on which recommendations will be issued on which cipher and device should be chosen to better fit the tasks at hand.

- **Chapter 7** consists of the second and final part of the practical assessment, this chapter contains the study of the transmission standards used within a IoT and seaport scenario, to determine how power efficient they are, as well as their available transmission rate. Further tests include how effective each standard's range is and study how they drop off with increased distance. Lastly, the scenarios established in Chapter 6 will, once again, be the subject of study to understand which standard's characteristics better fit with each requirement proposed.

- **Chapter 8** recaps all the information acquired throughout the conducted

practical assessment and informs the reader about all the decisions made, justified by their appropriate findings specific to each chapter of the thesis.

- **Appendices** contains all the complete average outputs for each metric and a collection of diagrams that, although contain interesting visualizations about each cipher's performance, were not included in Chapter 6 to avoid an overly extensive analysis.

Chapter 2

Background

This section intends to provide context and practical information on the topics that will be discussed throughout this thesis. Section 2.1 gives an introduction to security and the basic notions of encryption, hashing, the difference between a software and hardware implementation, and different hashing constructions, Section 2.2 and its subsections explain the most important terminologies of the technology used to conduct the studies, which is centered around symmetric encryption. Section 2.3 limits the usage of encryption to lightweight solutions, and its subsections include possible algorithms that will be subject to examination and testing. Section 2.4 describes the tools necessary in terms of resource-constrained devices and how they differ from each other. Section 2.5 approaches the transmission standards that are usually used in a lightweight context, and which will be tested later on for the thesis objectives, providing their specifications and regulatory compliance across different countries. In Section 2.6 we discuss the necessary structure and equipment to obtain the metrics's values. Emphasis on the power metrics, which have three possible choices as suitable candidates to extract this data. Finally, in Section 2.7 an overview of quantum cryptography and its current studies is provided; this is an important addition to the document, as this technology has the potential to render the findings of this thesis inoperable.

2.1 Protecting Information

Data usage and transmission are at an all-time high, however, with their expansion comes a certain level of risk. Attackers can compromise information and their security, targeting either individuals or businesses and causing disruption in its transmission or gaining unwanted access to it. To achieve security and mitigate the risk of a successful attack, three crucial cornerstones should be assured, **confidentiality**, which means securing data and only offering access to authorized entities, thus preventing unwanted disclosure of information; **integrity**, which focuses on ensuring data is not tampered with, either accidentally or on purpose, and remains in its original and intended form, allowing for both non-repudiation and authenticity; **availability**, which assures the access to information, for any authorized user, whenever needed [Popescul, 2011].

2.1.1 Using Encryption

To achieve confidentiality, we have encryption. In this process, an agglomerate of information is scrambled so that only authorized parties can decode it and understand its meaning. In more technical terms, it consists of transforming a plaintext understandable by everyone into a codified indiscernible construction, also known as cyphertext, which appears to be random.

2.1.2 Using Hashing

Beyond encryption, there is also hashing available in situations where confidentiality is considered necessary, whilst integrity and potentially authenticity are. It operates by, given an input with variable length, creating a hashed output with a fixed length which serves as a unique identifier of the original document, as depicted in Figure 2.1, hence attesting to its integrity. By hashing the document after arriving at its destination and comparing the resultant output with the original hash sent, a receiver can know for sure if the document was unchanged, as different outputs will confirm tampering [Tareq Hammad et al., 2017].

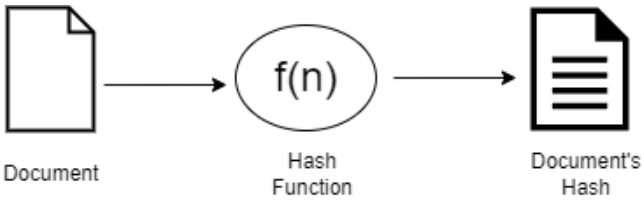


Figure 2.1: Description of a Hash Function operation.

Ideally, any sufficiently good hash function must abide by three properties:

- **Preimage resistance:** Given a hash value $H(m_0)$, it should be computationally hard to find any input message m_0 such that h = H(k, m_0), with H representing a hash function and k the key value.
- **Collision resistance:** Given two input messages m_0 and m_1, it should be computationally hard to find a hash value such that $H(m_0)$ = H(k, m_0) = H(k, m_1), with H representing a hash function and k the key value.
- **Second preimage resistance:** Given an input message m_0, it should be computationally hard to find a different input message m_1 such that H(k, m_0) = H(k, m_1), with H representing a hash function and k the key value.

2.1.3 Key, round, and block size

When talking about encryption algorithms, three important aspects of them, which will be discussed throughout this work, are the key, rounds, and data blocks. Key size, which is usually presented in bits, is correlated with the generic strength of the cipher, which, to be safe, should always be 128 bits or larger. This is done

to give strength to the algorithm against brute-force attacks, which are attempts at getting the correct key by guessing the entirety of bits in a key, which computationally is quite demanding and, in the case of a strong key, impossible for today's technology. Usually, the block and round size are more ambiguous and work together in their specific algorithms, and a higher value does not always equal more security, although it should still be considerable. For example, AES, a well-used encryption cipher, uses either 10, 12, or 14 rounds and a block size of 128 bits. Another important notion is that the number of rounds heavily depends on their security, AES can use 10 rounds because each one is complex, other algorithms use more rounds which are less complex to achieve comparable security, so it also depends on the round's complexity and design.

2.1.4 Software and Hardware Implementation

These two terms are discussed when deciding in which way an algorithm should be implemented. Whilst a software implementation describes an algorithm that is versatile and can run on a multitude of different systems and processors, a hardware implementation is made specifically for a single device, and takes advantage of its physical construction to run the cipher. When it comes to choosing a version, both have their pros and cons, more specifically a software implementation is more flexible given that the algorithm does not depend on the hardware used. Furthermore, it is also easier to modify and upgrade it since no physical connections need to be changed whilst making adaptations to the algorithm. However, when it comes to hardware implementations, these are faster, given that they do not depend on other services and have a circuit optimized to run the cipher. They are also more secure, given that it is much harder to tamper with a hardware implementation compared to a software one, mainly because the latter can be modified with more ease without raising any alert, unlike the former which, with any attempt to modify it would be noticeable given its directly implemented in the hardware; furthermore, there are specially modified chips that, in case of an attempt of access or tampering, will self destroy to prevent such modifications. Another reason for the increased notion of security stems from the smaller complexity of the chip, with fewer background programs the likelihood of a side-channel attack potentiated by another process is substantially reduced when compared to software implementations, which usually run on Central Processing Units (CPUs) with a higher complexity and several other processes, most of which unrelated to the algorithm. [Alkalbani et al., 2010]

2.1.5 Sponge and Merkle-Damgård Construction

When it comes to hashing, two main structures are used: Sponge Construction and Merkle-Damgård Construction. This subtopic aims to introduce these concepts and briefly go over how they work.

Merkle-Damgård Construction

In Figure 2.2 a simple description of how a Merkle-Damgård construction operates is depicted. We have an initial input message m that is divided into t parts, like m = (m_1 + m_2 + ... + m_{t-1} + m_t), a fixed and publicly available Initialization Vector (IV), a compression function f, and a H value that will hold the IV and subsequent result of each function's output. The process works by recursively combining each length-fixed block of the initial message with the H in the compression function (in the first round H_0 = IV). Afterward, the resultant product of this operation will be the new H (H_n = f(H_{n-1}, m_n), n = 1, ..., t) for the next round, merging with another block of the message in the next compression round. This process is repeated until the entire message is processed and the resultant outcome is the hash value for the message.

This construction is widely used, in the now unsafe MD4, MD5, SHA-0, SHA-1, and the safe SHA-2, although no structural vulnerability was found on the construct itself with any known cryptanalysis.

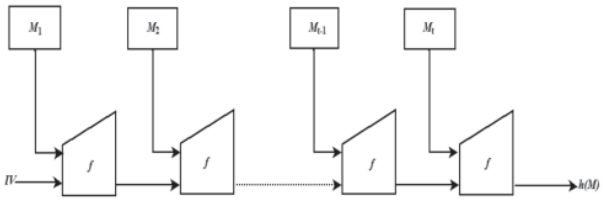


Figure 2.2: An example of hashing using a Merkle-Damgård Construction [Tiwari, 2017].

Sponge Construction

In Figure 2.3 we have a depiction of how a sponge construction works for hashing. The most important notion is that it is divided into two states, "Absorbing" and "Squeezing", hence its name. Firstly, in the absorbing phase, a portion of the message, with length r, is combined with small chunks of the initial buffer (also with length r) data using an XOR operation. Afterward, the mixed value plus the remaining buffer, which has r+c length, will pass through the function f which will semi-randomize the buffer. This process is continuous and will only stop once all the input data has been processed and inserted into the buffer. Once that occurs, the squeezing phase begins. The process is similar but with each iteration, instead of XORing data, the output of r size is removed from the buffer, whilst still passing through the function each time to randomize the remaining buffer [Bertoni et al., 2011].

The Sponge Construction was chosen for the SHA-3 hash function, the successor of the widely known SHA-2. Although performance and security standards appear to be similar between both functions, this construction tackles one important factor, most algorithms increase their output size to reduce the possibility of collisions, at the expense of performance, however with Sponge Construction (SP)

it is possible to mitigate this and keep a smaller state size without compromising the risk of collisions.

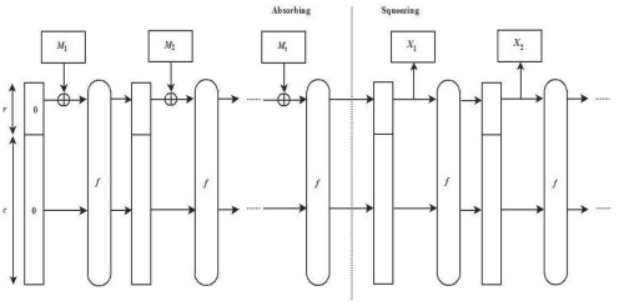


Figure 2.3: An example of hashing using a Sponge Construction [Tiwari, 2017].

2.2 Symmetric Encryption

Whilst encryption can either be symmetric or asymmetric, this study will focus on the former, given the necessity for higher resource computations for the latter, which are incompatible with the lightweight needs of this thesis. Symmetric encryption consists of using the same key, called a secret key, to perform both the encryption and decryption stages. This key is shared between the entities involved in these two stages so that only they can access the information being transmitted.

2.2.1 Block Ciphers

Block ciphers are considered a cornerstone of protocols' design for a large portion of all shared-key cryptography. They consist of a block of data that constitutes the plaintext which is treated as a whole and produces a cyphertext of equal size, with each block taking a variable amount of bits, usually between 64 and 128. Besides block ciphers, there are also stream ciphers, which differ in some ways: they always contain a secret "state", like memory, that is continuously updated whilst performing the encryption phase, and take advantage of streams of bits rather than blocks, performing encryption one bit at a time and disregarding the next bits in the process [Biryukov, 2004].

2.2.2 Substitution Permutation Networks

A Substitution Permutation Network (SPN) is a type of block cipher that has as input blocks of the plaintext and the key and applies alternating rounds of both substitution boxes (S-boxes), responsible for replacing the block values according to a rule or set of rules, and permutation boxes (P-boxes), which keep the values present in the block but change their order, all to produce the final block of ciphertext [Kong et al., 2015].

Two advantages of this method are the decryption process, simplified given that the operations need to be reversed to obtain the original plaintext, and their simple structure, making security analysis and understanding each step of the encryption process simple. When it comes to lightweight algorithms, AES and ASCON are two examples of an SPN structure.

Figure 2.4 depicts an example of an SPN round schedule with substitution, permutation, and key mixing, all applied in three subsequent rounds.

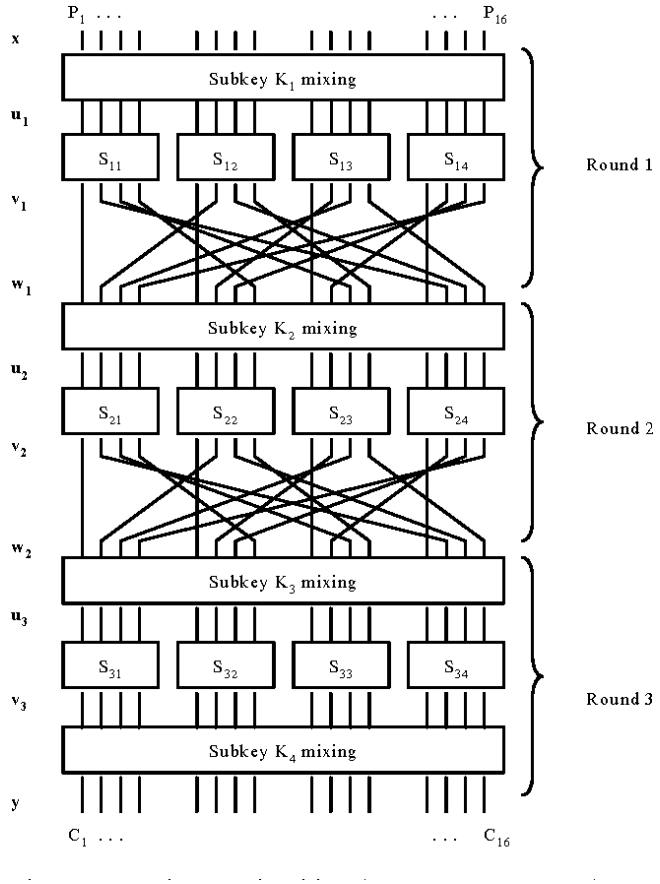


Figure 2.4: A three-round cipher using an SPN structure [Sakallı et al., 2005].

2.2.3 Feistel Networks

This second type of block cipher operation, known as Feistel Networks (FNs), splits the input data into two equal blocks, right and left, which are then processed in rounds of the encryption algorithm and interact with each other throughout the process. Similarly to SPN, the decryption process is also an advantage of this structure, given its similarity to the encryption process, plus a reversal of the key schedule [Kong et al., 2015].

The following image showcases a round of the cipher XTEA, which uses a Feistel Network to separate the block into right and left parts, computing several processes between them until the cipher text is achieved.

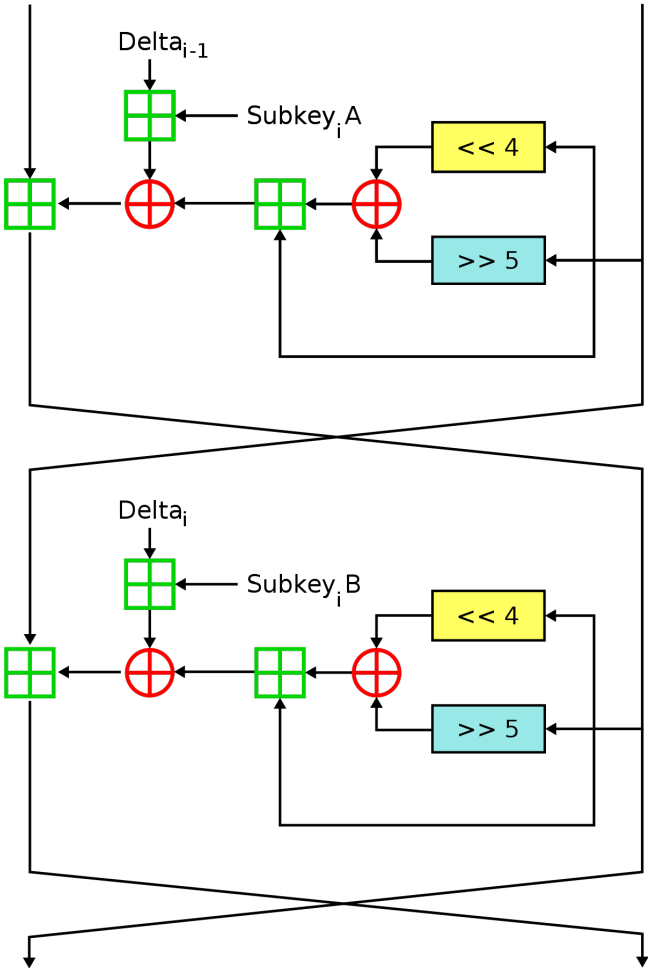


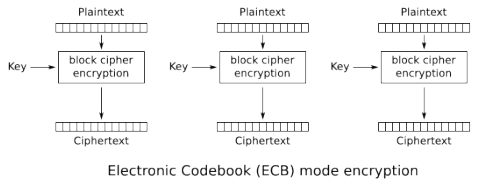
Figure 2.5: Sequence of block transformation using a Feistel Network [Hunn et al., 2012].

2.2.4 Modes of Operation

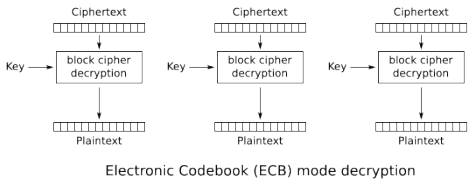
One further component of block ciphers is their mode of operation. This will dictate how the blocks of information are handled to transform the plaintext into a cyphertext, and vice-versa. Three main modes of operation will be subject to study throughout this thesis: Electronic Codebook (ECB), Cipher Feedback (CFB), and Galois/Counter Mode (GCM).

Starting with ECB, this one is the most basic of the three, dividing the message into blocks and performing both encryption and decryption on them separately, as demonstrated in Figure 2.6. Although known to be extremely fast and lightweight, given its simplicity and ability to parallelize both encryption and decryption given that these operations do not rely on the previous blocks of information to continue these processes (each block is independent and has no correlation with the next sequence of blocks), it possesses some security question marks, as it can leave some patterns while encrypting similar data, thus compromising its pseudo-randomness.

CFB on the other hand increases the overall complexity of the mode of operation and adds an important component, an IV that contains a block of bits that will



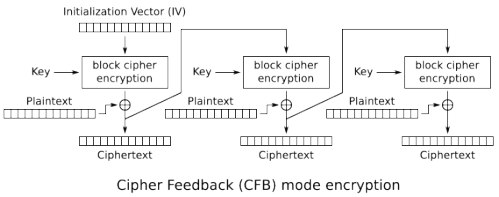
(a) ECB Encryption



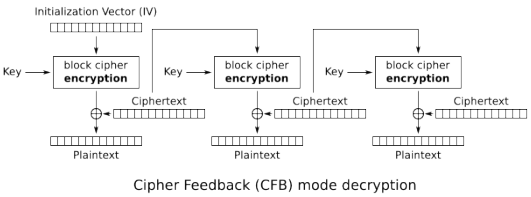
(b) ECB Decryption

Figure 2.6: Encryption and Decryption while using ECB [White].

randomize the encryption process and guarantee that similar plaintexts do not have any characteristic similar to each other after being encrypted, improving on the drawback from ECB. Figure 2.7 demonstrates both the process of encryption and decryption, which are similar but not equal. Starting with the encryption, the IV is used together with the key and the encryption method, and this product is then XOR-ed with the plaintext to generate the first block of ciphertext. After this, the process repeats itself, with the exception that the IV is now the final product of the previous round, repeating itself until the last block is encrypted. Decryption however does not pass the final product of each block to the next one, instead the block of ciphertext is passed. Due to this, decryption can be parallelized, given that, to reconstruct the plaintext, there is no dependency from one block to another. Given the sequential dependency on behalf of the encryption, this operation cannot be parallelized.



(a) CFB Encryption



(b) CFB Decryption

Figure 2.7: Encryption and Decryption while using CFB [White].

Lastly, there is also GCM which adds more complexity and additional data so that it can also provide authentication. Operations like encryption and decryption can be parallelized, improving the efficiency of any algorithms that use GCM. However, even with parallelization, given the ability to authenticate data, they are quite slow, especially when compared with ECB. An IV is also used to ensure pseudo-randomness and eliminate any common links between similar plaintexts that could otherwise compromise security. Figure 2.8 depicts how the encryption process is carried out using GCM.

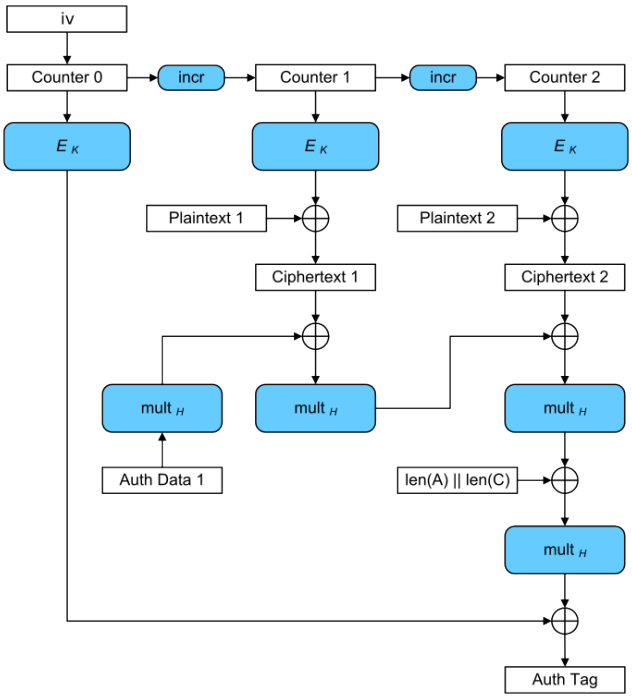


Figure 2.8: Encryption using GCM [McGrew and Viega, 2004].

2.3 Lightweight Approach

Necessary to meet the current demand for efficiency and energy saving, lightweight technology is on the rise, powered by resource-constrained devices that can achieve similar performances to other devices while operating with fewer resources. But how can this be put in numerical terms, when it comes to encryption? One article mentions that one of the criteria for a lightweight approach needs to have an implementation area between 1000 GE [1] and 3000 GE, anything below that would be considered ultra-lightweight and only to be used on devices with extremely limited computational capabilities and anything over that starts to leave the realm of a lightweight implementation. The number of GE used heavily depends on the implementation and can vary between two different algorithms of the same cipher [Meng and Buchanan, 2020].

2.3.1 Lightweight Encryption Algorithms

As the world accelerates toward a more resource-saving methodology, dominated by smaller and much more efficient devices, lightweight encryption serves as a security complement. One common misconception associated with lightweight encryption algorithms is that these are associated with shorter key and block sizes and thus are less safe than traditional ciphers, and whilst some of them do apply

[1] Gate Equivalence (GE), corresponds to the area occupied by an equivalent number of 2-input NAND gates [Banik et al., 2016]

this trend, most of the lightweight algorithms base their security on a reliable and safe design rather than reducing these parameters to values not considered safe [Mouha, 2015].

Their application has been steadily increasing throughout the years, ranging from sensors to gateway controllers, and companies are understanding the efficiency potential of these ciphers, which comes from their small power consumption, which allows for more savings and batteries that can last longer. The latter is especially important in contexts where the device is not easily accessible (i.e., a sensor on top of a tower to measure weather metrics) and any operation to either change the battery or the device itself has a bigger cost.

The following algorithms that will be presented were selected based on their security value, trustworthiness, and performance and will be subject to further examination to determine which ones will be part of the comparison study on IoT devices.

AES

AES is a lightweight encryption algorithm, published in 2001 by NIST, which consists of a block cipher with an SPN format, 128 bits of block size, a variable number of rounds between 10 and 14, and also a variable key size, which can either have 128, 192, or 256 bits. Some of its advantages are its enormous trust capacity against all types of attacks and its robustness as an encryption protocol. One possible drawback stems from the fact that since it allows for a higher key size, some impacts on performance and power consumption might be noticeable [A. Dawood and I. Hammadi, 2017].

ASCON

ASCON is another lightweight cipher that was developed in 2014 and is based on an SPN format, thus using block ciphers. It famously won the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) competition in 2023 for authenticated encryption mechanisms that are lightweight, can yield high performance, and have very sound security and defense in depth. Looking for more technical details, it has 128 bits of key size, up to 128 bits of block size, although 64 and 128 are the recommended values and 6 or 8 rounds for encryption and decryption.

As for its advantages, ASCON is suitable for resource-constrained devices given its relatively smaller blocks and key sizes, as well as a comparatively smaller state size and low implementation area compared to AES. These, combined with the relative ease of implementing from scratch on both hardware and software, make this cipher a good choice. However, it also has a disadvantage given that it is not parallelizable on a message block level, cannot take advantage of some of the high-performance implementations that are available for its competitor AES [Dobraunig et al., 2021], and, given that it was published later than most of the mentioned algorithms, like AES, CLEFIA, HIGHT, and more than 2 decades after

both TEA and XTEA ciphers, it lacks the amount of testing that comes with the lifespan of a cipher whilst in the public domain.

CLEFIA

CLEFIA is a block cipher, that uses Feistel Network, and was developed by Sony and subsequently published in 2007. It supports three possible key sizes, each with 128, 192, and 256 bits of strength, 128 bits of block size, and three possible number of rounds, namely 18, 22, and 26. The main goal behind its development was to obtain good security, speed, and cost for implementation for an expanding market of file distribution. This was achieved by, at the time, being the first block cipher employing the Diffusion Switching Mechanism (DSM) to increase their resilience and immunity against differential and linear attacks. Furthermore, by using lightweight components, efficiency is guaranteed for both software and hardware implementations, thus reducing the cost per encryption or decryption cycle [Sony, 2007].

HIGHT

This cipher was first published in 2006, shortly after being adopted as an ISO standard under the code ISO 18033-3:2010. Its key size is 128 bits, being complemented by 32 rounds and a block size of 64 bits, with each round having four parallel Feistel functions. According to their creators, what sets this cipher apart from the rest is its performance results, which exceed the newly proposed lightweight versions of AES, whilst maintaining a strong level of security [Hong et al., 2006].

LBLOCK

LBlock was published in 2011 as a lightweight encryption algorithm based on block ciphers, more specifically, a Feistel structure, with a key length of 80 bits, 64 bits of block size, and 32 rounds. This algorithm, claimed to by their authors, at the time, outperforms other lightweight block ciphers on a hardware implementation and also achieves good performance for software implementations on an 8-bit microcontroller, which is the usual configuration for resource-constrained devices like smart cards and sensors [Zhang and Wu, 2011]. However, according to the submission criteria for new lightweight authenticated encryption algorithms published by the National Institute of Standards and Technology (NIST) in 2018, the acceptable key size is now 128 bits, with an attack complexity of at least 2^{128} [NIST, 2018], which would put LBlock below these specifications. Although the cipher was published 12 years ago, according to these standards its security is not optimal, which is a drawback to its usage for encryption.

SIMON

SIMON is a lightweight block cipher based on Feistel Network, published in 2013 by the National Security Agency (NSA). It has a wide range of combinations for its key size, ranging from 64 to 256 bits, block size, ranging from 32 to 128 bits, and rounds, between 32 and 72, depending on the level of security and performance that one wants to achieve. The most commonly used combinations have a key size of 96 or 128 bits. A characteristic that makes SIMON stand out above similar lightweight ciphers like AES or HIGHT is its performance, with a significantly lower area of implementation and, concurrently, lower power consumption, as well as its versatility of combinations of key size, round size, and total rounds, allowing for implementations in constrained devices that otherwise cannot use encryption [Alkhzaimi and Lauridsen, 2013].

TEA

TEA is another block cipher that was first published in 1994, standing out as one of the oldest and most used lightweight ciphers. It has a key size of 128 bits, 64 bits of block size, and a variable number of rounds, 64 being the recommended value on which a variety of cryptanalysis was conducted over time. It operates on a Feistel Network, with every two rounds equating to one cycle. For its time, and also today, it stands out given its simplicity of implementation, fast execution times, and usage of minimal space [Virmani et al., 2013], however, its susceptibility to chosen plaintext and related key attacks, combined with a maximum of 128 bits of key size, make it a weaker choice, especially when compared to AES or the more recent ASCON. Three years later, it was replaced with XTEA, an improved version that limited the impact of these attacks on the cipher.

XTEA

This cipher was published in 1997 and consists of a Feistel Network with a variable number of rounds, usually using 64 as it is the recommended value to ensure general safety, 128 bits of key size, and 64 bits of block size. The algorithm appeared as a replacement for TEA by the same authors, which was necessary given TEA's simple key schedule and weakness against related key attacks, thus representing a security issue [Moon et al., 2002]. XTEA's new structure builds on its predecessor's problems, using a more complex key scheduling process and a rearrangement of the operations applied to the two boxes.

Other Lightweight Approaches

Beyond the already mentioned lightweight algorithms, there are a few that were also considered for analysis and possible inclusion in the testing phase, namely both light and extra light versions of DES, ICEBERG, KTANTAN, LED, mCrypton, PRESENT, QARMA (as well as modified versions of these latter two algorithms that improve on their performance), PICCOLO, RECTANGLE, Sato, SEA,

SFN, and SLIM. There were two reasons for not including these ciphers in a dedicated subtopic in this chapter, firstly they did not have any available implementations that we could find for all Internet Of Things (IoT) devices (except for SLIM), and secondly, because marginal analysis conducted on their security, performance, peer-review, and usage in resource-constrained devices were subpar when compared with the other encryption ciphers. One point that should be mentioned however is that SLIM did have an implementation available and its security and performance appeared to be good, however, it was released in 2020 and very few cryptanalysis has been done on it, unlike ASCON which was also published quite recently in 2023 but won the CAESAR competition, having been subject to far more scrutineering.

Comparison of the Algorithms

In Table 2.1 an overview of the main attributes of each lightweight encryption algorithm is provided to establish a comparison ground between them. The chosen criteria are the design of the cipher, key size, block size (both expressed in bits) and number of rounds.

Algorithm	Design	Key Size (bits)	Block Size (bits)	Nº Rounds	Gate Equivalent (GE)
AES	SPN	128, 192, 256	128	10, 12, 14	2400
ASCON	SPN	128	64, 128	6, 8	-
CLEFIA	FN	128, 192, 256	128	18, 22, 26	2488
HIGHT	FN	128	64	32	3048
LBLOCK	FN	80	64	32	1320
SIMON	FN	Most common 96 or 128	32-128	32-72	763
TEA	FN	128	64	Recommended 64	2100
XTEA	FN	128	64	Recommended 64	2000

Table 2.1: Comparison of Encryption Algorithms' Specifications.

2.4 Resource-Constrained Devices

With the rise of IoT technology, efficiency is in an all-time high demand, which created a market for devices that have computational capabilities to perform successfully the tasks they have to perform and networking abilities that allow their connection with the Internet and surrounding devices. So, to minimize production costs and adapt their products to the tasks required, these controllers are equipped with low-power embedded computational devices that are severely constrained in their processing capabilities and perform their work with less power consumption, thus contributing to smaller costs and improved efficiency [Sehgal et al., 2012].

2.4.1 Arduino

Arduino is an open-source microcontroller that provides an easy platform to develop code via its Integrated Development Environment (IDE), which compiles

C++ language and is easily erased and reprogrammed on short notice. It is based on several microcontroller boards and can work as a resource-limited computer, ideal for performing relatively easy tasks that do not require much computational power. Furthermore, it has the capability to be connected to multiple sensors and provides a straightforward approach to analyzing this data through its IDE, which is important for the area of study that revolves around IoT technology [Louis, 2016].

From a technical standpoint, the type of Arduino used to aid this research is an Arduino Uno, which is based on a main microcontroller called ATmega328P which has a frequency of 16MHz and uses 5V of electric potential. Furthermore, it has 2KB of Static Random Access Memory (SRAM), which means that data will be preserved as long as there is power, and 32KB of flash memory.

2.4.2 Raspberry Pi Family

Another IoT device commonly used as a resource-constrained device is the Raspberry Pi. It consists of a small computer that runs a Linux distribution and serves as a cheap and lightweight alternative. This device contains a high amount of computational power and is capable of performing various computing tasks and interfacing multiple devices [Nayyar and Puri, 2015].

Compared to Arduino, its power is greater, as it contains more resources to meet its higher computational capabilities, making it a larger and less versatile alternative. When applied in resource-constrained scenarios, they are usually used at a higher level, which means that they can take more calculations and data treatment than the Arduino, serving as an intermediary point between the data collected and subsequently sent to outside servers or databases.

For this study, three Raspberry Pi devices will be used, each with different technical specifications, so that we can have a device for each level of complexity subject to analysis.

- **Raspberry Pi Pico:** The more resource-constrained device, which should perform at a low level. Although not very complex, it is still comprehensibly faster than the Arduino Uno, with 125 MHz of clock speed, 264 KB of SRAM, 2 MB of Flash Memory, and a two-core processor. Similarly to the Uno, neither Wi-Fi nor Bluetooth are available with this device, so if there was a need to transmit data elsewhere, an external board would need to be used.

- **Raspberry Pi Zero 2W:** Far more advanced than the previous Raspberry Pi, has a graphical interface available a boasts a clock speed of 1 GHz, 512 MB of SRAM, a four-core processor, and an available board for both Wi-Fi and Bluetooth (version 4.1). This device is probably already outside the realm of resource-constrained and thus should operate at a much higher level, within the management of operations and processing of much more complex and significant flows of data.

- **Raspberry Pi 3B+:** The most complex device available for this study, also comes with a graphical interface and even more complex technical specifications, more specifically, 1.4 GHz of clock speed, 1 GB of available SRAM, a four-core processor and, once again, built-in support for both Wi-Fi and Bluetooth (version 4.2). This device is certainly not considered resource-constrained and, if used, should operate at a high level, handling considerable amounts of data and information.

The three devices establish a step curve in complexity and clock speed, allowing for much higher coverage of all needed services within the thesis objectives and use-case of seaports.

2.4.3 ESP Family

The third (and final) category available for this study is the ESP family which, for this document, is comprised of an ESP32 and ESP8266, two low-end devices that are somewhat resource-constrained and operate at a lower to medium level of operations. They were designed by Espressif Systems to provide low-power and low-cost alternatives to the market of IoT devices. Furthermore, they can be programmed in a variety of languages, however, within the context of this thesis, they will be used in conjunction with the Arduino IDE, which provides an accessible and simple communication with the hardware.

The ESP32 is the successor of the ESP8266 and has a higher level of complexity, with a clock speed of 240 MHz (80 MHz for ESP8266), 520 KB of SRAM and 4 MB of Flash Memory, against the 64 KB and 4 MB of ESP8266, and a dual-core processor, while the latter only uses a single-core processor. ESP32 also has a built-in board for both Wi-Fi and Bluetooth (version 4.2), while ESP8266 only has a Wi-Fi board so any Bluetooth communication would require the usage of external hardware.

2.4.4 Device Comparison

In Table 2.2 we have a comparison of all of the devices' technical specifications based on the content displayed in the previous sections. As noticeable, Arduino Uno, Pico, and ESP8266 are the more resource-constrained devices, given their relatively low memory and low clock speed, and should be useful to find any potential limitations within the lightweight encryption techniques or transmission standards used. Then, ESP32 stands on a medium level of complexity; given its characteristics, it should handle both encryption and data transmission with ease, although certain parameters like speed, power, and memory consumed are key to correctly assess and classify the device's performance. Lastly, both Zero and 3B+ are noticeably faster and more complex than any of the other alternatives and should handle the necessary data for encryption and transmission with relative ease, although, given the high performance expected, certain details like power consumed could be detrimental for their usage in the stipulated context.

IoT Device	Frequency	SRAM	Flash Memory	Cores	Wi-Fi	Bluetooth
Arduino Uno	16 MHz	2 KB	32 KB	1	-	-
ESP8266	80 MHz	64 KB	4 MB	1	Y	-
ESP32	240 MHz	520 KB	4 MB	2	Y	Y
Raspberry Pi Pico	125 MHz	264 KB	2 MB	2	-	-
Raspberry Pi Zero 2W	1 GHz	512 MB	SD CARD	4	Y	Y
Raspberry Pi 3B+	1.4 GHz	1 GB	SD CARD	4	Y	Y

Table 2.2: IoT hardware technical specifications.

2.5 Transmission Standards

Given that this thesis aims to contribute to the seaports, it is logical that we study how the data can be transmitted and evaluate its protection, to ensure resilience against potential attacks. There are several ideal candidates that can both perform in constrained environments and have drastically different operational levels regarding range and signal intensity. This thesis will provide background into Wi-Fi, LoRaWAN, Bluetooth, and ESP-Now given that, after performing the literature review and analyzing the technologies used in both seaports and IoT technology, they were mentioned several times and regarded as important to establish communication between the different devices.

2.5.1 Wi-Fi

Wi-Fi is an agglomerate of wireless network protocols, which started being commercialized in 1997 and allows devices to exchange data using radio waves. The main goal of Wi-Fi is to allow a high throughput data transfer, within a small range (usually up to 1000m but heavily dependent on several aspects, such as its implementation, location, materials in the way, and version used). The usage of Wi-Fi can be done by having a router or another device that can manage the data traffic and communicate with any nearby devices, however, it also supports the creation of an ad-hoc [2] network. Furthermore, another important notion in an ad-hoc network is the notions of single-hop and multi-hop. Whilst in the former a singular device carries all the information and acts as a management point, the latter utilizes intermediary devices to increase the range and allow for farther away devices to still be able to connect to the controlling node, thus increasing the usual 100m range of communication to a theoretical value above 1km. With the increased range, a multi-hop configuration has also higher latency, it is more complex to implement given the variety of intermediary nodes to manage and is also associated with a higher energy consumption, natural with the extension of the Wi-Fi reach [Klimiashvili et al., 2020].

When it comes to regulatory legislation and its frequency usage, Wi-Fi typically

[2]Without the usage of a wireless base station, namely a Wi-Fi router or aWireless Access Point (WAP), a singular device can replace them and be responsible for exchanging data in the network with the various existing nodes. It is easier to implement than a full Wi-Fi structure with a WAP and is mostly used as temporary networks or in cases where the range needs to be increased and extending any cables or adding new equipment is not feasible [Froehlich, 2022]

operates in the 2.4 GHz and 5 GHz frequencies, however, its usage is still possible in the sub-GHz spectrum, as well as 3.6 GHz, 6 GHz, 45 GHz, and 60 GHz. These latter frequencies and their channels are defined by each country and have to abide by specific power output limits and a multitude of restrictions that can limit their output, range, and whether they can be used indoors, outdoors, or both.

2.5.2 LoRaWAN

LoRaWAN is a recent wireless technology from 2015 that operates unlicensed bands under 1 GHz, with no licensing cost for its usage, proposed by the LoRa Alliance, which is designed to optimize data transmission in Low-Power Wide-Area Networks (LPWAN). It works with a gateway device, which receives data from up to thousands of edge devices and subsequently can transfer that information to the internet. According to its specifications , the range which is covered by this technology can be between 2 and 5 Km in urbanized spaces and up to 15 Km in less populated areas, although this heavily depends on the chosen Spreading Factor (SF) [3] which not only contributes to the range of transmission but also interferes with the data rate, which in LoRaWAN's case, can go from 0.25 kbps to 50 kbps, depending on the chosen SF. Furthermore, as expected this contributes to a tremendous power saving when compared to other standards [Klimiashvili et al., 2020].

One more important thing to point out about LoRaWAN is its regulation across different countries. For example, in China, the information is transmitted on the frequency bands between 470 MHz to 510 MHz, however, when looking at North America, this value is greater, between 902-928 MHz. Furthermore, the number of channels allowed, the data rate, and the output power also change between regions. Whilst in North America, 80 channels exist with data transmission between 1 kbps and 22 kbps and a maximum output value of +30 dBM, in Europe only 10 channels are available, between 0.25 kbps and 50 kbps, with a lower maximum power allowed of only +14 dBM. It is also noteworthy that for regions with heavy industrial demands like China, Japan, and India, no regulatory values beyond the frequency band are in place, although they are currently under evaluation and a final specification should be issued in the meantime [Odunlade, 2019].

2.5.3 Bluetooth

Bluetooth is a wireless technology, first introduced in 1998, that provides short-range connectivity alternatives for small portable devices. Throughout the years, five different versions have been launched, to increase their performance metrics and fix blatant security concerns. Nowadays, most devices operate on Bluetooth

[3] The SF controls and adapts the transmission and power rates for the device, allowing for several optimizations. A higher SF is equivalent to a lower transmission rate but, at the same time, a higher range. This value can be set as an Integer from 7 to 12.

Version 4 or 5, with the former introducing Bluetooth Low Energy (BLE), a novel feature specifically designed to support IoT devices with limited resources and battery life, that otherwise can't benefit from this technology . When compared to similar low-power alternatives like ZigBee or Wavenis, BLE has been the most adopted standard given that it is significantly more energy efficient than any of its competitors due to a streamlined protocol stack and an already established brand in the Bluetooth Special Interest Group (SIG) [Al-Shareeda et al., 2023].

When compared to Bluetooth Classic (v4.2), BLE, in theory, only delivers a third of the maximum transmission speed (3 vs 1 Mbps) and the same maximum range of about 50-60 meters, although this value can vary greatly depending on the device, implementation, and other environmental factors, however, the biggest difference between the two is the huge power efficiency which could be reduced by 90% versus Bluetooth Classic, extremely essential to ensure devices with low or costly accessibility can use the same battery for several years while not compromising their data transmission. A Bluetooth Version 5 was launched in 2016 with an extended range of up to 300 meters and a maximum transmission rate of 2 Mbps when in low power mode and 50 Mbps in standard operations (these improvements were possible thanks to the increase in packet size to 255 bytes) however, none of the devices present in this study have access to this new version, only operating in Version 4.1 or 4.2 [Nick He, 2024].

Lastly, both Bluetooth Classic and BLE operate on the unlicensed but regulated ISM band, with an established frequency between 2.4 GHz and 2.48 GHz, which has 79 channels of 1 MHz available. Furthermore, it is built on the principle of Frequency Hopping Spread Spectrum (FHSS) that is responsible for regularly changing the channel used for Bluetooth communication.

2.5.4 ESPNow

ESP-Now is another connection-less protocol developed by Espressif Systems to ensure low-power, "built in-house", transmission standards for their IoT devices, without resorting to Wi-Fi. Each ESP board needs to pair first with the other devices being used, after that, the connection established is persistent, which means that if one device disconnects, it will automatically connect once again to ensure that communication continues [Urazayev et al., 2023].

The standard currently supports both encrypted and unencrypted communication, with a payload of up to 250 Bytes being carried out. It supports two types of communication, namely:

- **One-Way Communication:** Data is exchanged in a unidirectional way, with one (or more) ESP devices sending data to one or multiple devices. This type of communication is called a master-slave relationship, a master sends data to a slave device.
- **Two-Way Communication:** The ESP devices exchange data in both directions, and each device is ready to receive and send data. It can be done

between two or more devices, which can lead to a network of ESP boards forming.

Furthermore, when it comes to the available range, depending on the type of mode used for ESP-Now, it can reach 500 meters, although it demands quite a lot of power to achieve these distances and most packets will not reliably be captured by the other device and thus lost. On the lower-power version, the range should be reliable close to the 150-200 meters mark, anything beyond that will result in packet loss.

As for the frequency bands used, ESP-Now operates on the 2.4 GHz band, although the channel chosen must be taken into consideration as it might receive interference from other Wi-Fi networks currently in use nearby.

2.5.5 Comparison between the standards

In Table 2.3 we have a brief comparison of the key points between these five technologies, namely their estimated range of operation, the data rates that each can ensure at a minimum and maximum, the average power output expressed as a qualitative value and their frequency usage, expressed in GHz. Highlighting what was previously said, these five standards and their usage are highly dependent on the environment in which they will be inserted. The power saving that BLE brings is extremely useful in the context of IoT technology and, more specifically, resource-constrained devices that do not have long battery life and constantly exchange data. Furthermore, if the data is small enough, LoRaWAN with its extremely large range of up to 15 km would be the ideal option, however, if the data needed to transmit is excessively large or needs faster transmission rates, ESP-Now or Wi-Fi would be the preferred choices. The only use-case in which Bluetooth Classic has the upper hand when compared to BLE is its data rate however, it is hard to justify choosing the former given the latter's unattainable power-saving, especially when, for a much faster rate and slightly more power consuming, Wi-Fi could be chosen instead.

Standard	Range	Data Rates	Power Usage	Frequency Band
Wi-Fi	100-1000 m	10 to 100+ Mbps	Medium	2.4 or 5 GHz
LoRaWAN	up to 15 km	0.25 to 50 kbps	Low	Below 1 GHz
BLE (v4.2)	up to 60 m	1 Mbps	Very Low	2.4 to 2.48 GHz
BT Classic (v4.2)	up to 60 m	3 Mbps	Low	2.4 to 2.48 GHz
ESPNow	up to 200 m	1 Mbps	Low	2.4 GHz

Table 2.3: Transmission standards technical specifications.

2.6 Tools to obtain the metrics

Another important step is to consider the possible instruments in use to gather all the necessary data. Since time, speed, and algorithm characteristics can be

measured using coding practices, knowledge, or tools/libraries, we only lack a power-measurement device that can yield the energy results.

2.6.1 Usage of a USB Power Meter

The first approach and arguably the easiest way to conduct the testing process would be to use what is called a USB Power Meter, which consists of a device that is connected between the computer and the hardware which will measure both port current and voltage, along with some other interesting statistics like average consumption over some time. It is much simpler and displays the same information as the two next-circuit options, although a potential negative downside is the relatively few authors discussing this as an option for conducting power measurements with IoT devices. Figure 2.9 showcases the connection between the USB Power Meter and both the power supply (which will be a computer) and the two IoT devices.

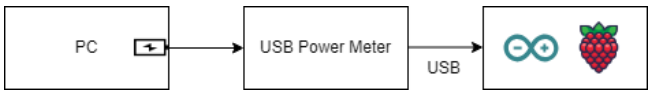


Figure 2.9: High-level view of the USB Power Meter usability.

2.6.2 Usage of a Multimeter

This testing approach has been widely documented with multiple real-world applications to measure power consumption on IoT devices and consists of using a device called a multimeter to measure the current of a closed circuit, in this case, to calculate the power used by each algorithm's execution. Furthermore, both a load and a breadboard must be used since the 5V connection that a computer provides to the device can't be measured in this way (in theory they can but it would involve cutting and splitting the cable used for the connection, which would be far too complex and permanently damage the cable(s) in question). Once the setup is built, probably similarly to Figure 2.10, the value will be displayed on the multimeter's monitor.

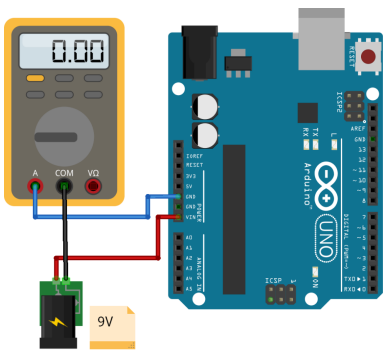


Figure 2.10: Real-life example of a multimeter used to measure power consumption [Diy IOT, 2021].

2.6.3 Usage of a Sensor

This third option would be similar to the previous scenario, only instead it would remove the multimeter and implement this sensor in its place. This however would be more complex since it would require a second Arduino or similar device that would read and display the values measured in the serial monitor. Once again, for the same reason listed previously, it is not possible to measure the power consumption of the IoT device using the USB cable to supply the power, so a power supply would be needed. In Figure 2.11, we have represented the schematic of the connection between the second Arduino and the INA219 sensor. To complete the circuit the Vin+ would need to be connected to the positive side of the power supply and the Vin- to the positive side of the device. Furthermore, the negative side of the supply would connect to the Ground pin on the IoT device.

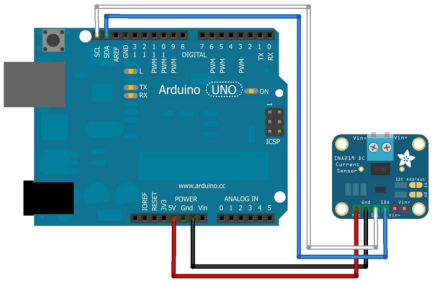


Figure 2.11: Second Arduino connection with the INA219 sensor to output information [Shah et al., 2019].

The previously described circuit connection would thus look like Figure 2.12; a breadboard would also be required, as well as some cables to complete this process. It is also noted that the left device which is represented as an Arduino can also be any other hardware, as the goal is to measure the power consumption of this particular device.

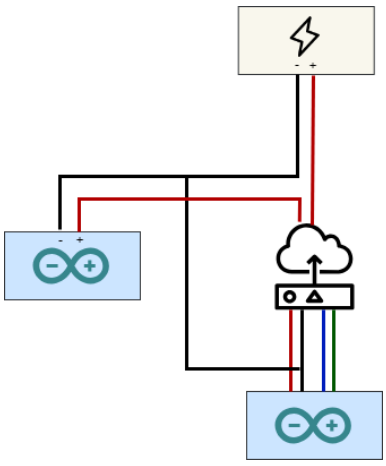


Figure 2.12: Entire circuit for power measurements.

2.6.4 Multimeter vs USB Power Meter

These two devices are widely used in the context of measuring power data, especially in the IoT context, so the objective of this chapter is to compare their technical characteristics and their easiness of usability.

Whilst the digital multimeter usually has a current measuring resolution of 0.001 mA to 1 mA the USB power meter has 0.01 mA to 1 mA. The displayed value is exact, but the multimeter usually refreshes the displayed information faster than the USB Power Meter (250ms vs 500ms). Of course, this information is also dependent on the type of digital multimeter available to conduct the study. Another important characteristic is the logging capabilities since framing exact time windows is important to separate the encryption and decryption portions; on the previous topic, a USB Power Meter had this capability, although multimeters are usually also able to record this data for analysis. Furthermore, the Arduino and Raspberry Pi power consumption, when idle, averages 0.2 Watts for the first and 1.4 for the latter. This yields a current measure of 56 mA and 280 mA respectively, which should be a big enough value so that both power-measuring devices can detect subtle changes when running different algorithms. With this data in mind, we believe that when within the same price range, a USB Power Meter can have the same specifications as a multimeter, with the added facility of its usage and implementation.

2.7 Quantum Cryptography

Quantum computers are currently being studied to develop practical applications and, when commercially available, should yield much more computational power than any traditional device. Right now, this technology is not there, but the potential is huge. This, of course, will have an impact on existing encryption algorithms, since with more speed and efficiency, the easier the cipher is to break. However, to fully break a cipher, quantum devices would need an extremely large amount of qubits [4] to be successful, which should not happen in the near future. When this technology is present, due to how quantum technology operates, traditional encryption will be heavily compromised. The main thesis is that there are two algorithms, Shor's and Grover's, which possess different characteristics that can aid an attacker in breaking a cipher.

Starting with Shor's algorithm and staying within the context of asymmetric encryption, this algorithm is extremely efficient in solving discrete logarithms and integer factorization problems, which are the basis of some existing ciphers like Diffie-Hellman and RSA. The first logical step to consider would be simply to increase the length of the key; however, this will not work as a sufficiently large key that could reasonably withstand Shor's algorithm would simply be too time-consuming to perform any encryption in a reasonable amount of time. Could

[4] Qubits are the equivalent of classical bits but, instead of only having two states like bits (0 or 1), they can still be represented as a combination of both. These qubits represent the strength of the hardware

they be replaced with other algorithms? Yes and no, yes because there could be potential candidates to be developed that could withstand the capacity of problem-solving displayed by quantum devices but no because such solutions are still not available, with any traditional one already developed suffering from a lack of efficiency that invalidates its use.

Another relevant approach, and probably more within the context of this work, would be Grover's algorithm, which is applied to symmetric encryption ciphers, which in quantum technology allows for a quadratic speedup of key searching, allowing a much faster brute-force attempt to retrieve any key. Putting this in numerical terms, any key available today when faced with quantum technology would see its strength reduced by half, searching the available space with a time complexity of $O(2^{(n/2)})$, with n being the key size in bits, instead of the classical $O(2^n)$. Taking the algorithms approached for the lightweight encryption, most of them would not yield the necessary security of the now 112-bit key standard, set to increase to 128 by 2030, as most of them only offer up to 128-bit key size, mostly due to efficiency. One possible alternative could be the AES-256, based on a 256-bit key size which, with quantum technology, would see its strength cut to 2^{128}, which is still within the established security limits. This would, unlike the case with asymmetric encryption, still offer a reliable, safe, and somewhat efficient traditional algorithm that could be used in a quantum scenario; however, most likely, new algorithms would need to be developed much sooner to offer more efficiency and power saving to resource-constrained devices, which would have a harder time or simply not work with current ciphers using 256 or higher bits of key size. One more key point of Grover's algorithm is that it can also be used against hash functions such as those listed previously, with the available solution being to double the size of the hash output to offer a similar security result when compared to pre-quantum computing.

How does this concept insert itself into the IoT environment? With this future threat, cryptography that can withstand such attacks must exist and be progressively deployed and benchmarked, however, given the resource-constrained nature of many IoT systems, an increase in complexity and resource-demanding could strain even further the available performance and severely hinder its functionality. Studies conducted point out that at this moment, most of the existing quantum-resistant ciphers demand more resources than what is possible of these devices, with only a small minority being able to deliver a lightweight approach compatible with IoT scenarios [Liu et al., 2024]. Furthermore, there is also a lack of standardization for newer and more resource-constrained approaches to quantum cryptography that tender to the requirements of lightweight devices.

To summarize, quantum cryptography is still not here, however, due to the investments in research it is safe to assume that modern standards for encryption ciphers won't be enough for most of them, which is why it is extremely important to also invest in new algorithms that can either be safe and efficient with higher key sizes for symmetric encryption and which are based on problems not easily solved with quantum technology, this for asymmetric encryption [Schmid, 2023]. Once new standards are established, the forecast is that new advancements in computational power and ciphers' security should be linear, as it is today with

the traditional and commercially used algorithms.

Chapter 3

State of the Art & Literature Review

This section presents the state-of-the-art, including the literature review of the topics that align with the thesis objectives. Section 3.1 will introduce the reader to the literature review and outline the methodology and strategies on which the entire chapter will operate. In Section 3.2 we provide an overview of the existing IoT technologies that are currently being developed or already in place to optimize seaports. In Section 3.3 use cases of USB Power Meter, a very important device to measure power consumption to optimize energy usage, are analyzed, more specifically which devices are being used and for what purpose. In Section 3.4 we go over the studies conducted on lightweight encryption, essential to relate with our own study, and their algorithm selection, metrics used to evaluate and compare them, and their respective conclusions.

3.1 Methodology for the Literature Review

3.1.1 Research Questions

Before conducting work in any field, we should be aware of the level of knowledge already at present and try to identify possible areas where our contribution will be valuable to the scientific community. Thus, the goal of the systematic review should be to answer the following research questions concerning the state of the art of the chosen topic, to summarize what is already available and where this work can improve it:

- What IoT technologies and devices are used in seaports, what purpose do they serve, and what type of data do they transmit?
- What comparisons have been made between which algorithms and which metrics were used to measure their performance?
- Have newer lightweight algorithms done more to increase security/performance or consume fewer resources?

As highlighted previously, these three questions will allow us to understand both the current information on seaports, IoT technology, and lightweight encryption, to give a generic overview of the solutions already in place and identify work fields that are currently missing knowledge.

3.1.2 Preliminary Results

Keywords used

To conduct the initial search, some terms must be used to facilitate the reach of the necessary papers. The following combinations present in Table 3.1 were used in the search engines presented in the following subsection.

Nº	Keyword Combination
1	"lightweight encryption" AND "IoT" AND "seaport"
2	"lightweight encryption" AND "IoT"
3	"lightweight encryption" AND "Arduino"
4	"lightweight encryption" AND "Raspberry Pi"
5	"IoT" AND "seaport"
6	"comparison" AND "lightweight encryption"

Table 3.1: Keyword Combination for paper searching.

Databases researched

Next, the databases with scientific papers from which the results were retrieved based on the search queries mentioned above. These will provide the number of articles found, as well as the papers to be selected for the systematic review process based on their characteristics and how well they address the topic.

- Google Scholar
- Scopus
- IEEE Xplore

Each platform would then yield a numerical value for the total results of papers found, which will then be subject to exclusion criteria.

Papers and documents found

The usage of these platforms yielded many results, to be more specific, 12 864 papers of all kinds were obtained using the search queries on the research databases, although some of these are not unique as a lot of the queries yield identical papers in their reported numbers. From analyzing table 3.2, Google Scholar by far provides the most results, followed a long way back by both Scopus and IEEE Xplore. These papers share in some way or another some (or all) the core characteristics of the thesis and thus will be important whilst conducting the systematic review.

Nº	Google Scholar	IEEE Xplore	Scopus
1	3	0	0
2	4390	104	243
3	433	1	3
4	595	3	9
5	2530	5	26
6	4370	119	30
Total	12321	232	311

Table 3.2: Total number of papers found pre-exclusion.

3.1.3 Research Strategy

Criteria for exclusion

Whilst 12 864 papers were found within the context of the search, a majority of them need to be excluded to narrow down the scope of the review. Although the thesis is being developed with some recommendations for seaport in mind, the papers that did not reflect approaches that used encryption or relevant IoT technology using low-resource consumption and optimizing efficiency and resource saving were excluded as this is a cornerstone of the research study. Another important step of this exclusion process is based on the top results, especially within Google Scholar, given that it yields a far larger number of articles when compared to the other databases. Newer articles were preferably chosen as they had newer encryption standards (and thus more aligned with the lightweight encryption ciphers) and their top results were compared based on their titles and abstracts. This last portion of the analysis was also used with Scopus and IEEE Xplore to greatly narrow down the list of papers chosen. After this process, 10 papers were selected given their compatibility with the study and comparison of lightweight ciphers, and their work will be reviewed and compared per the strategy below in the following topics of this chapter. The papers will be presented in the following order:

1. Realtime Measurement Integrated Hydrodynamic Conditions For Improving Port Performance
2. Marine Radio for Voice Communication System on Very High Frequency (VHF) Spectrum
3. Internet of Things for Smart Ports: Technologies and Challenges
4. LoRaWan Capacity Simulation and Field Test in a Harbour Environment
5. A Power Model for Monitoring Environmental Parameters on the Edge
6. Performance evaluation of Attribute-Based Encryption on constrained IoT devices
7. Performance Evaluation of Lightweight Encryption Algorithms for IoT-Based Applications

8. IoT Security: Data Encryption for Arduino-based IoT Devices
9. Lightweight Cryptography
10. Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platform

The first four papers are related to IoT technology in seaports, the following three concern the usage of USB Power Meters with resource-constrained devices to get power consumption data, and the last three articles detail studies already conducted on lightweight encryption and their methodology.

Strategy for building the literature review

Now that the papers to conduct the systematic review have been identified, they should be summarized and compared according to their composition and how well they address and give information that can be used in the topic under work. Firstly, a short description of what they address should be made, followed by comparing their content and findings with similar papers, to understand which methods are preferable. For the encryption algorithms some possible criteria to look for are the number of algorithms chosen for the solution/comparison, and, subsequently, the number of metrics used and what field they address (i.e., more related to performance or security).

3.2 Seaport applications of IoT technology

This first topic of the state-of-the-art provides context into the used IoT technologies in seaports, depicting real-world solutions that are integrated to optimize its performance and/or security. The main goal here is to assess the used devices and transmission standards, based on lightweight solutions only, as well as their usability and purpose. The output should give an overview of the most important findings of each paper analyzed, taking into consideration the IoT usage that can be useful for our work.

3.2.1 Identified Works and their contributions

Paper 1 - Realtime Measurement Integrated Hydrodynamic Conditions For Improving Port Performance

The first studied article [Ganjar et al., 2019] intends to apply real-time monitoring solutions to aid seaports with traffic management and damage mitigation/avoidance. This is done by developing a connected and efficient network of buoyance devices, which are able to measure three crucial parameters: wave fluctuation, tidal, and flow measurements. The first point is connected to the variance of height experienced by the device whilst being in a wave and can be

measured with buoy sensors to determine said fluctuation. Tides can be measured with ultrasonic sensors that emit ultrasonic waves that give real-time feedback on the status of the tide and, finally, flow measurements which are achieved via speed and two ultrasonic sensors (to guarantee redundancy and accurate results), that accurately determine the speed and direction of the flow observed by the sea. All of this is powered by an Arduino Uno microcontroller, although research conducted in 2013 described a similar solution using a microcontroller ATMEGA 162.

With the data gathered it is then sent to a Raspberry Pi via the Wi-Fi protocol so that it can be unpacked and read correctly, giving feedback on the current sea situation to the seaport. One particular aspect of this paper that was important was the comments made under the conclusion section, in which the authors mention gaps in graphic data due to the exhaustion of power to the tide gauge, which can be of interest for the necessity of lightweight encryption for minimal resource IoT devices.

With this, the full scope of the data can be used to make decisions on specific routes towards and from a seaport that is optimal and avoids rough seas, thus minimizing the risk of an incident and optimizing operations.

In summary, the IoT devices used in this solution are:

- Arduino Uno microcontroller
- Raspberry Pi
- 3x Ultrasonic sensors
- Speed sensor
- Accelerometer and Gyroscope sensors (although not mentioned in the paper, it is possible they are used to map out wave height).

Paper 2 - Marine Radio for Voice Communication System on Very High Frequency (VHF) Spectrum

The second paper in question [Hariyanto et al., 2019] focuses on a common problem in Indonesia regarding voice transmission to other boats and ports. Dating 2019, a large percentage of all boats, especially fishermen, can't communicate via the appropriate frequency (several of them use aviation channels instead of the maritime ones) or do not communicate at all. The authors propose a simple, widely available, and cheap solution to solve this issue, which consists of a device that performs voice communication on the Very High Frequency (VHF) spectrum.

The core of the device is an Arduino Pro Mini (3,3V / 8MHz) chosen because of its relatively easy usability with a custom IDE and being open source. Furthermore, the device is also equipped with a wireless voice transceiver, DRA 818 V, that is capable of both receiving and sending transmissions wirelessly. The transceiver

is also capable of performing modulation on the sound waves, more specifically Modulation Frequency.

This paper aims to solve several problems in underdeveloped countries, allowing for exclusive radio channels for communication so that positions can be transmitted (particularly useful in emergency cases) as well as report several pieces of information useful to the seaport such as weather and sea conditions, potential piracy, or enemy vessels nearby, crucial to ensure the harbor's good operations and increased safety for all ships.

Paper 3 - Internet of Things for Smart Ports: Technologies and Challenges

This third paper [Yang et al., 2018] is probably the most complex and modular IoT solution of all studied papers, as it contributes to the state of the art of IoT in seaports at a multitude of levels, instead of a singular application to solve a particular problem. The authors describe the goal as a "smart port", whilst attempting full automation of data transmission which can facilitate the seaport's operation ability and wirelessly transmit data from various sensors to control facilities.

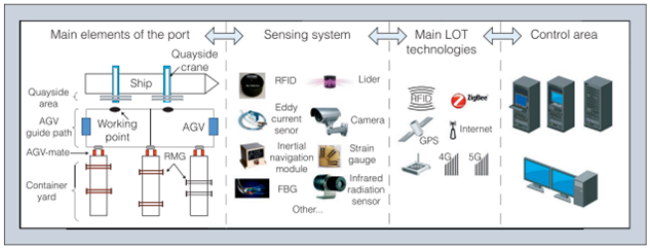


Figure 3.1: High-view layout of the IoT technology needed for the smart port [Yang et al., 2018].

As demonstrated in Figure 3.1, there would be a need for several different devices to achieve automation. The goal is to achieve autonomous management of a cargo terminal, by removing and loading cargo as well as placing and driving it to the appropriate stations. From what we were able to gather, the core of the paper intends to study and analyze the different solutions for sensing and transmitting data necessary to achieve the autonomy of the port, comparing them in regard to their pros and cons. Firstly, they do an overall comparison between their smart port and a manual counterpart, highlighting how efficient, environmentally friendly, and cost-saving their approach is. Then, they go over the core necessities to implement their thesis, first comparing different sensing technologies to monitor the Structural Health Monitoring (SHM) of the cranes and metal structures, to find potential fatigue or micro-cracks before they cause any damage. In Table 3.3, the authors make the comparison between the technologies currently used for this monitoring, highlighting that the Strain gauge is by far the most used one, quoting the authors: "Translating the structural strain into resistance change is inexpensive and stable." Inductive eddy current sensors and ultrasonic sensors are described as the two go-to online solutions to the SHM task, with the last one easily applicable in an IoT scenario to detect inconsistencies in

the acoustic impedance and thus detect potential cracks in the material.

Type	Parameter		
	Cost	**Range**	**Applications**
Strain Gauge	Low	High	Stress and strength of metal structures
FBG	High	High	Stress and strength of metal structures
Current sensor	Medium	Short	Detection of flaws, like cracks and deformation
Ultrasonic Sensor	Medium	Short	Detection of flaws, like cracks and deformation

Table 3.3: Sensing technologies used for SHM. Adapted from [Yang et al., 2018].

Moving on, proximity sensing technology is also needed to avoid collisions and allow the cranes and nearby operational vehicles to gather any container. In Table 3.4, four sensing technologies are again compared, with the ultrasonic sensor remaining as a common point. In the paper, the author does not emphasize which technology is preferred, as some have different applications based on their cost, range, and environmental adaptability. From our point of view, ultrasonic sensors could be chosen for both this and the previous SHM to simplify the complexity of the smart port and still yield good automation.

Type	Parameter		
	Cost	**Range**	**Applications**
Ultrasonic Sensor	Medium	High	Anti-collision for crane
Laser & Lidar	Medium	Very High	Anti-collision for crane and other components
Infrared Sensor	Low	Short	Anti-collision for crane
Electromagnetic Sensor	Low	Very Short	Anti-collision for crane and other components

Table 3.4: Distance measuring sensors for the smart port. Adapted from [Yang et al., 2018].

There is also the question of autonomous navigation and how a smart port is able to support it. Table 3.5 contains the most used solutions in this regard. The first two are needed throughout the most critical phases of the port's operations, especially when handling loads by the cranes, whilst the last two serve noncritical roles due to their relative position and lower accuracy.

Type	Parameter		
	Cost	**Accuracy**	**Position measurements**
GPS system	High	Very High	Absolute
Laser-based navigation	High	Very High	Absolute
Encoders	Low	Low	Relative
Inertial navigation	Medium	High	Relative

Table 3.5: Navigation sensors for the smart port. Adapted from [Yang et al., 2018].

Beyond this, other IoT devices are used like cameras and laser sensors to safeguard and ensure continuous monitoring of the containers and nearby systems. Cameras help with the identification of said containers and certify the correct alignment with the autonomous structures whilst their location in the crane or transport vehicles is achieved by laser sensors.

In Table 3.6, some of the wireless technologies available to transmit all the necessary information between the IoT devices mentioned were included. ZigBee is a newly presented alternative in the IoT scenario, while other proven technologies like Wi-Fi or 4G/5G also have practical applications in this field. Furthermore, the paper also demonstrates the application of wireless sensors in the SHM process to find fatigue in the structure, all sending their data to a coordinator via the below ZigBee protocol.

Wireless Technologies	**ZigBee**	**Wi-Fi**	**RF**	**4 G**
Speed	250 kbps	300 Mbps	9.6 kbps	100 Mbps / 1 Gbps
Relative Cost	Low	Medium	Medium	Medium
Frequency	784 MHz	2.4 GHz / 5 GHz	433 MHz	$1700-2100$MHz $2500-2700$MHz
Range (outdoor)	100 m	100 m	20 km	-
Compatibility	IEEE 802.15 .4	IEEE 802.11 ac / n	802.11 ac	LTE

Table 3.6: Wireless technology used for the smart port. Adapted from [Yang et al., 2018].

Paper4 - LoRaWan Capacity Simulation and Field Test in a Harbour Environment

Finally, the last paper in question [Victor et al., 2018] is particularly interesting for the topics in question since it addresses both resource-saving and seaport efficiency. It has practical applications in Denmark and aims to increase efficiency by using IoT technology to automatically point out available spots in the harbor for ships to be placed in, instead of personnel doing physical searches to find out which spots are available or taken. To achieve this, the authors propose an Arduino-based solution (Seeeduino), that establishes communication via the LoRaWan protocol (an extremely lightweight communication method that allows long-distance data exchange and very high power saving, at the cost of small data payloads) and uses an ultrasonic sensor to detect the boats in the nearby spaces. This device would need to be installed in all parking spaces for boats so that the information can be gathered and processed. It also takes advantage of a gateway that receives all information that subsequently uses the Message Queuing Telemetry Transport (MQTT) protocol (also a lightweight communication protocol that publishes information according to a certain topic and can even ensure data retention in case of downtime). With this, all that is left is a server that can host this information and both display and store it.

This device network demonstrates the benefits of IoT technology when applied in the seaport scenario, particularly in solving a very niche problem related to quickly identifying vacant spots for incoming vessels. The usage of protocols such as LoRaWan and MQTT, as well as guaranteeing intermittent data transmission only when a change happens, further demonstrates the power-saving capabilities of IoT networks.

Furthermore, the paper also mentions a prototype solution to verify that the LoRaWan technology will be accessible even to the spots farther away from the

gateway. To do so, an Arduino was used in conjunction with an RN2483 PICtail/PICtal Plus Daughter Board. To test this, the prototype was carried throughout the testing harbor (a port in Svanemoellehavn), to confirm said distances, with success.

In summary, the IoT devices used in this solution are a Seeduino and Arduino microcontrollers, paired with an ultrasonic sensor for presence detection. Both of which are powered by LoRaWan and MQTT protocol solutions.

3.2.2 Type of data sent

All of the described solutions use IoT technology on a seaport context to exchange data crucial to achieving the desired performance, which is why this subtopic will focus on the type of data transmitted in each paper, allowing us to understand if the information packets are larger or smaller and thus garner crucial knowledge of the operation of this common technology on the studied use case so that we can replicate the same packet size for our studies and align them with real-world examples.

- **Paper 1:** The authors again are not very conclusive on the details of the transferred data, but from reading the paper it is possible to estimate that at least 2 values are transmitted, a float with two decimal places that represent the temperature measurement and an integer that gives information on the tide. Although not explicit in the paper, other integer or float values might be sent as well for speed measurements of said waves.
- **Paper 2:** In this paper, the only information transmitted is voice data in the VHF spectrum between vessels and ports.
- **Paper 3:** This paper is once again very complex, but we can deduce some values. The transmission rate to achieve automation would need to be extremely high, however, for most tasks requiring coordination, only precise location data for control of cargo/crane or values of stress for the SHM monitoring would need to be sent, meaning only some float or integers values.
- **Paper 4:** The simplest of all papers studied, would only need a single bit to transmit whether or not a parking spot is occupied. Although not mentioned, this could be the approach used.

3.2.3 Comparison and Analysis

After having studied these and more papers searching for IoT solutions in seaports, their devices, and the purpose of their application, some conclusions can be drawn:

- Most of the papers, excluding paper 3, described solutions for a specific problem that could be solved with IoT technology, and most of them resorted to Arduinos attached to specific sensors needed to automate their

scenario. Throughout this research, it was hard to find ports or solutions that integrated a Raspberry Pi (except paper 1 and a few other solutions), with Arduinos usually taking up most of the tasks needed.

- Arduinos were most of the time used to gather the data via their sensors and transmit it elsewhere, whether to a specified gateway or a Raspberry Pi in paper 1. A wide range of sensors was used as part of the IoT structure, with each purpose clearly defined in their paper's explanation.
- Paper 3 stood out given that it implemented a much wider solution to fully automate an entire cargo seaport, providing multiple IoT-powered solutions that aim at providing full automation.

The following table 3.7 overviews the main IoT technologies identified throughout the state of the art on this topic, highlighting their problems and solutions:

IoT Device	Papers Used	Purpose
Arduino	1	Extract Weather Data
	2	Voice Transmission via VHF
	4	Transmit information about parking spot for vessels
Raspberry Pi	1	Receive weather data and store it

Table 3.7: Summary of the IoT devices used in the paper solutions.

3.3 IoT scenarios which use USB Power Meter

Another important piece of this section concerns the USB Power Meter. From the research conducted, we determined that this device is very useful for gathering power-related data from resource-constrained devices, which is why it is helpful to know the different approaches of this technology coupled with IoT devices to understand how these measurements are performed, as well as comparing different available models.

3.3.1 Identified Works and their contributions

Paper 5 - A Power Model for Monitoring Environmental Parameters on the Edge

In this paper [Hurbungs et al., 2021], the main objective is to measure the power usage and requirements for resource-constrained devices, more specifically a Raspberry Pi and two Arduinos, which work in the context of edge computing, where all the data is processed and calculated close to their source and subsequently transmitted from the Raspberry Pi to a SQL database. The document discusses several approaches to obtaining power metrics, such as PicoScope, SD3004, and the INA219 (already discussed previously), however, the authors argue that these might not be practical to deploy in large IoT environments given their complexity

to measure this data (a high number of connections necessary, and more power drawn). This led them to adopt the USB Power Meter to register these values, more specifically, a UM34C. In Figure 3.2, a graphical description of the testing environment is showcased, with the two Arduinos gathering the necessary data from the sensors, which are then connected to the Raspberry Pi responsible for processing the data and sending it to an outside database. This demonstrates the core function of edge computing, where data is processed closer to the source to minimize the data transmitted and thus optimize its transfer speed. Furthermore, the UM34C power meter is connected between the power source and the IoT device, to register the fluctuation in power consumption over the testing period of the solution.

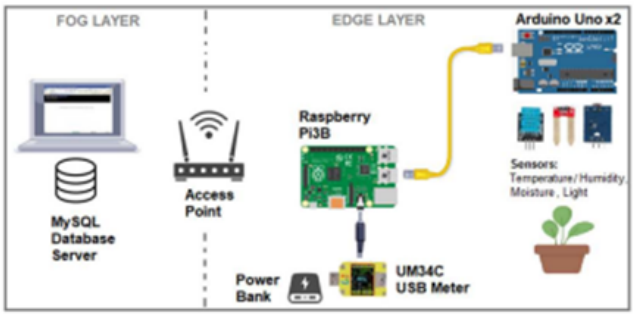


Figure 3.2: Testing environment for the edge computing evaluation [Hurbungs et al., 2021].

To start the testing campaign, the authors measured the three IoT devices' power consumption whilst idle, which returned 0.28 Watts and 0.56 Watts for the two Arduinos and 1.4 Watts for the Raspberry Pi. Although the paper doesn't go into detail on this topic, the power measurement in both Arduinos was probably gathered by connecting the USB power meter (and respective power supply) directly into them to measure this value. To emulate the stress that the Raspberry might experience whilst in operation, a command in Linux (stress-ng -c 4 -l percentage –timeout 60s) was used, with the value percentage referring to the value for the load the system will endure, for 60 seconds. The idea is to increment this value consecutively by a factor of 10 until it reaches the maximum load and, at each step, measure the power consumption of the device. This test showed that it increased linearly throughout the test.

Another important landmark of this paper is to validate their results, which means that the authors want to prove that their readings with a USB Power Meter are valid. For that, they used a hardware-based approach to calculate the power usage of the IoT devices, which involved knowing some information about the number of cycles used (obtained via the command: (iostat -t -x 1) and calculating the value based on a formula. The results were very promising, with a Mean Squared Error (MSE), Root Mean Squared Error (RMSE) very close to 0, and an R-squared of nearly 1, indicating a high accuracy of the data. The following Table 3.8 contains the power consumption of both Arduinos measured with the Power Meter and using the hardware-based approach.

Power	USB Power Meter (W)	Hardware-based approach (W)	Accuracy (%)
Arduino 1	1.68	1.71	98.24
Arduino 2	1.96	1.91	97.45

Table 3.8: Comparison between USB Power Meter and Hardware-based.

Paper 6 - Performance evaluation of Attribute-Based Encryption on constrained IoT devices

The article [Perazzo et al., 2021] analyzes the performance of different Attribute-Based Encryption (ABE) algorithms to enhance the authentication process on encrypted data and consists of the usage of public keys. This comparison will be made on IoT devices, namely ESP32 and RE-Mote, to understand how much power they draw and whether they are compatible and have good usability on resource-constrained devices. The most important feature is the usage of a USB Power Meter to carry out the measurements.

Going over the methodology, it is possible to draw several comparisons with the work carried out in this document. The encryption ciphers will be implemented and will run on a sequential logic, measuring the time statistics for each step of the process (key setup, encryption, and decryption). Furthermore, to carry out the power measurements, the authors describe the usage of a high-precision USB Power Meter, the AVHzY USB Power Meter Tester, which has increased accuracy and resolution when compared to the more used meters (0.0001 A and 0.0001 V) and has an extremely useful property, which saves all the data measured on a computer, allowing for its posterior access and analysis. This is extremely useful for the paper and our topic since it gives the possibility to match the exact readings with the exact timestamps measured for each step of the algorithm testing campaign.

The paper then goes on to compare the different algorithms and how they can be used on the different IoT platforms, as well as providing information about their useful battery life when used for encryption or decryption. However, the biggest takeaway from this analysis is the usage of a power meter that can log the data it collected for analysis. In our specific case, this would be extremely useful, as it can allow us to accurately detect which parts of the algorithm usage take up more or less power, how these can be optimized, as well as get extremely accurate values for the power consumption (resolution of 10^{-15} Megawatt Hour (MWh)).

Paper 7 - Performance Evaluation of Lightweight Encryption Algorithms for IoT-Based Applications

The article [Panahi et al., 2021] is perhaps the most important since it heavily relates to the work done for this document. The work aims to explore the best lightweight algorithms to perform encryption and decryption on an IoT device before transmitting said information. To achieve this, several metrics were used to compare them, namely energy consumption/throughput, memory usage, and

execution time on two devices: Raspberry Pi 3 (the same device as used in our work), and an Arduino Mega 2560. The paper goes into detail explaining the characteristics of each algorithm used in the testing campaign, like key size and number of rounds. Furthermore, a comprehensive list of related works that describe similar testing campaigns of lightweight ciphers on other resource-constrained devices (with specifications of the metrics used) is also showcased.

The testing campaign is comprised of two devices, one which encrypts the data with the target algorithm and transfers it to a second device, responsible for decrypting it. The transfer occurs with Dropbox, with metrics time being calculated at the device to remove upload and download times from the equation. The Arduino is powered via USB cable by a computer, whilst the Raspberry Pi receives its power from a 10 400 mA Power Bank. As for the power consumption, a USB Power Meter was used, although the model is not specified. The metric used for this step is Energy consumed in Joules, which means that they first need to calculate the Charge, which is obtained by multiplying the current reading (in Amperes) by the time of encryption or decryption and then multiplying the resultant value by the Voltage, also measured by the Power Meter.

3.3.2 Comparison and Analysis

After summarizing and reading the selected papers on the current knowledge in this area, we can understand how important the USB Power Meter is for power measurements. Table 3.9 provides an overview of the different models compared by their voltage and current resolution (sensitivity to change voltage and current measurements), the capabilities to log the information on a device, extremely important to depict the exact timeframe where the encryption and decryption happened and then be able to establish an average value for their power consumption and their relative cost.

Paper	Power Meter	Voltage Resolution	Current Resolution	Logging to PC	Relative Cost
5	UM25C	0.001V	0.0001A	Yes	Low
	UM34C	0.01V	0.001A	Yes	Low
6	AVHzY Power Meter	0.0001V	0.00001A	Yes	Medium
7	KWS-V20	0.01V	0.01A	No	Very Low
	PAC1934	0.001V	0.001A	Yes	High

Table 3.9: USB Power Meters used with IoT devices.

3.4 Study of lightweight algorithm comparisons

In this topic, different papers that narrate the comparison of different lightweight encryption on resource-constrained algorithms are included. The main goal is to understand the setup efforts to carry out the comparisons, namely the implementations used, how to ensure each device has legitimate comparability between each other, the algorithms selected for this purpose, and, lastly, the metrics chosen to conduct the work. The expected output is a summarized version of each

paper providing the mentioned details, as well as a comparison between their selected ciphers and metrics.

3.4.1 Identified Works and their contributions

Paper 8 - IoT Security: Data Encryption for Arduino-based IoT Devices

The article [Abdullah et al., 2022] depicts the implementation of three encryption algorithms, Caesar Cipher, SHA 256, and AES-128 on an Arduino device, with the goal of contributing to the state of art on security within the IoT world, without the trade-off of efficiency. The objective is to secure data transmission between an Arduino and the user, or vice versa, more specifically password sharing. Firstly, an attack vector is described in which the usage of Ettercap, a sniffing application that can intercept packets in a network and, if necessary, modify them (Man in the Middle (MiTM) attack, Address Resolution Protocol (ARP) poisoning), will be necessary to test whether or not the data transmitted is protected or not over the Wi-Fi network used.

Moving on, the Arduino used is an Arduino Uno, with a Wi-Fi module that is able to send and receive information via the Wi-Fi protocol. Furthermore, the ESP8266 module is also used to enable the Wi-Fi capabilities on the device and is installed via the Arduino IDE, which is accessible when connecting the Arduino to a computer. The algorithms will also be run on the ESP8266 board, as well as the necessary code to transmit data. There will also be an Hypertext Transfer Protocol (HTTP) webpage put in place so that users can input data to be encrypted and choose the algorithm used to do so. This information will be relied on the Arduino, which will send back the information encrypted. It is also important to mention that the code will be run using a library available in the Arduino IDE (as mentioned before in the implementations subtopic of this paper) with the author mentioning that some adjustments had to be made to execute the algorithms properly without compiling errors, although he doesn't go into further detail on this regard.

The previously mentioned attack was run on two versions of this paper, one without encryption and another with encryption. The results, as expected, demonstrated that the second version would protect the data from being understood in all stages of transmitting post-encryption. The authors also compare the three algorithms in terms of their reliability, speed, and accuracy. One interesting result of the research topic is that AES-128 was not able to handle more than 100 characters for encryption, freezing the Arduino. However, we do believe that this will not be a significant issue, especially given the fact that the type of data transmitted in the seaport scenario is very small in most cases to maximize efficiency. AES also obtained the highest encryption speed of the three algorithms, and all had 100% accuracy, which was cross-tested with online tools to decrypt and compare the message and cipher text.

Overall, although this paper seems a bit incomplete and vague at times, it provides a good understanding of the hardware, modules, and libraries needed to

execute the algorithms and provides some good context into the performance of a cipher relevant to the research, AES-128. Furthermore, it also gives context on some possible metrics to compare the algorithms that might be useful to the future chapters of this thesis.

Paper 9 - Lightweight Cryptography

This paper [Rahman et al., 2022] has much greater detail than the previous one, and the main point to be discussed here is the methods' topic, in which the authors describe their choices and subsequent implementations. For starters, a Raspberry Pi Zero W and an Arduino Teensy 3.2 were chosen, similarly to this research, to benchmark the lightweight encryption algorithms. Their choice was derived from their popularity in the IoT context and the relative simplicity of their approach. One more important aspect mentioned is that as the Arduino is smaller and less powerful than the Raspberry Pi, is more suited to test the limits of the hardware when applied to encryption.

When it comes to the ciphers chosen, the authors mention the wider range of options for the Raspberry Pi versus the Arduino, given that it can run several languages and even high-level ones like Python which are more user-friendly, whilst the Arduino can only run C++ code from its IDE, although for the Raspberry Pi only lower level languages based on C or Assembly were considered to maximize efficiency. The Arduino used a library in its IDE that contains several algorithmic implementations, like AES and ASCON (which are already listed in the implementations subtopic). When it comes to the Raspberry Pi, to maintain the same algorithms' implementation and thus have measurable points of comparison in their security and performance, the authors used a C++ library called Piduino which allows for the usage of Arduino code and its libraries, mimicking the device, allowing for the execution of identical implementations on both devices.

Finally, another key aspect of our future research will be how can we define the metrics that are going to be used to benchmark the algorithms and assess their performance. This question can have a multitude of answers depending on the scope of the project; in this paper, they were looking to collect mainly time-elapsed statistics, with emphasis on the time required for each stage of the process (key setup, encryption, decryption). Furthermore, some temperature data, memory usage, and CPU utilization were also gathered, although only possible for the Raspberry Pi, given its capacity to run background processes. However, the data gathered in this regard was not considered meaningful because they had no way of establishing a comparison with the Arduino and had their own problems; for temperature, they would get a peak level after some runs that would stabilize until the end of the task, meaning they couldn't compare between runs the temperature difference, which was related to the uptime of the processor and not the complexity of the task at hand; for CPU and memory usage it would be necessary to measure each subroutine independently, which also did not happen since their usage was far below the device's limit and did not change substantially.

Paper 10 - Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platform

Finally, the article [El-hajj et al., 2023] also dives into comparisons between different lightweight encryption algorithms implemented on both Arduino and Raspberry Pi. For the latter, the authors use MinGW as a container for the GNU compiler (GCC) to compile their algorithms' code, since they will use low-level versions of them to eliminate any potential software barriers. The Arduino will compile the code via its IDE, as expected. As for libraries, for the calculations of some metrics (which will be discussed below), they resorted to a single file responsible for calculating them and, in the case of the Raspberry Pi, exported the results for a CSV file, with the help of a tool called PLX-DAQ. Note that given the simplicity of the Arduino, the CSV export cannot be achieved. When it comes to extra hardware, an Adafruit INA219 current sensor will also be used to obtain power-related data that will be used in the metrics, using a second Arduino to display this information on the serial monitor. One important takeaway from this paper for us would be the metrics chosen to compare the algorithms between them and they are defined in Table 3.10 below.

Tool Used	Measurement	Metric
Algorithm Specs	Key size	Bits
	Block size	Bits
	Rounds	N/A
Programming + Equation	Speed throughput	Bytes/s
	Speed latency	Cycle/Block
Power Sensor	Power throughput	Joules/s
	Energy latency	Joules/bit
Arduino IDE / Size command	ROM used	Bits
Arduino IDE / Valgrind	RAM used	Bits
Size occupied on memory	Code Size	KBytes

Table 3.10: Metrics used in the paper.

As for the algorithms used to test and compare, there were a total of 119 ciphers selected (some are the same type, but with different technical specifications like key and round size), across both traditional and new lightweight ciphers. This is important as it relates to one of our research questions which aims to provide some context about whether newer solutions have particular advantages and are better molded into the resource-constrained environment or if any security concerns may arise from their usage. However, this paper does not compare these two groups between each other and instead provides a comparison within each group, which does not allow for this direct comparison to be established. The study presents its results by giving a graphic depicting the 10 best-performing and the 5 worst-performing ciphers of each metric, which is necessary given the wide range of algorithms being tested. A comparison of the performance of both Arduino and Raspberry Pi is also provided, which concludes that the majority of the algorithms had better encryption and decryption speed and energy consumption on the Raspberry Pi, which is attributed to its "Hardware and Software Architecture", with most of the algorithms not following this norm using key

sizes not recommended, around 80-bits.

3.4.2 Comparison and Analysis

From the summarized papers we can understand that there are several approaches to compare each algorithm. Every research used an Arduino and most of them also resorted to the Raspberry Pi since these two devices are considered lightweight and thus ideal candidates to test the efficiency of these ciphers. Furthermore, both mention the usage of both time-related and power-related metrics as important measures to compare the algorithms and assess their performance. From these papers, we can already understand some ciphers that might have relevance to our study, namely AES, and ASCON (which appears to be an ideal candidate for newer lightweight encryption). As for the methodology, they also provide adequate information on how to conduct this study on IoT devices and, more importantly, specific scenarios that are also based on the transmission of data, like paper 8 with Wi-Fi, as this is a topic that will be useful for conducting our own testing schedule.

Chapter 4

Seaport Security and Performance Requirements

This chapter will provide a summary of the research conducted on seaports, combining both the data already acquired in the previous chapters and new information specifically tailored to the analysis of security concepts, attacks, and requirements that must be in place at all major seaports, on which IoT technology can have a considerable impact and adaptability.

4.1 Security in a Seaport Environment

With the growth of IoT technology, its attack susceptibility is also expanding. According to a study from 2017, a 600% rise in cyberattacks on this technology was observed when compared to the previous year [Jović et al., 2019]. In a paper published in 2015 [Gamundani, 2015], threats targeting IoT technology can be divided into four categories, namely:

- **Application-Based:** This category describes the overall security structure inherent to each application used in an IoT device, either internal or external. Threats that fall into this category are directed at the device with the aim of compromising its effectiveness. In a seaport context, assuming we have a surveillance system in place, both attacks that are internal and target its software and subsequent functioning or external which can tamper with the positioning of a camera or cover it are dangerous and can compromise the effectiveness of this system, allowing for unauthorized actions to take place.
- **Connection-Based:** This section is related to anything involving a network and, thus, addresses both the threats of a connection between two or more IoT devices and in the sending and receiving processes at each device. To correlate with the seaport scenario, data being transmitted by a sensor to a computational device like a Raspberry Pi, any potential attack that can compromise the security of the data exchanged at any stage of the information

transaction process falls into this category. Furthermore, the main countermeasure appointed in this scenario is the usage of encryption to establish a secure pathway for data to flow in a network.

- **Platform-Based:** Thirdly, this type of threat is directly connected with the application-based attacks mentioned previously. This section describes threats against a device with high-level permissions or a service, both of which are used by various nodes in a network and depend on them. With the expansion of IoT technology, some services are not keeping up with modern security vulnerabilities but, since many devices heavily depend on them for their functioning, they can thus become compromised and contribute to an alarming spread of a cyberattack in a network.

- **Other attacks**: Lastly, this category merges the notion of multi-level attacks that resort to a combination of threats based on the three previously mentioned categories to not only compromise a single node or a section of an IoT network but the entire service, rendering it inoperable.

The port of Rotterdam, the largest in Europe, has undergone a drastic modernization to include IoT services in its environment. With 140 thousand ships annually served, the usage of sensors and subsequent complementary devices like Arduinos and micro-controllers are compulsory to control the traffic flow and get weather, sea, and geographic data that can predict and optimize the arrival and departure times of each vessel according to the measured parameters in these categories. However, in 2017, a large cyberattack disrupted the operations at Rotterdam for several days, by rendering various IoT powered systems that handled cargo removal and transportation inoperable, causing a tremendous financial impact. According to [Yaqoob et al., 2017], "Key management is one of the crucial issues in cybersecurity and is more complex in the IoT, wherein many devices are resource-constrained". By this representation, encryption and key management processes were improved to mitigate the risk of similar attacks taking place in the near future, although the chosen solutions and improvements have to adapt to the lightweight necessities surrounding seaports. Furthermore, to increase the resiliency of the seaport, in a measure also applied in numerous other ports like Antwerp, and to mitigate Connection-Based threats, a protocol named FERM was adopted to raise awareness with any companies that operate within the vicinity of the seaport about the importance of cybersecurity, encourage hackers to discover weaknesses and report them to the seaport authorities, and any company that, again, is part of the ecosystem of the harbor, has to report any disruption to its services to the seaport, to mitigate the spreading of any security threat.

Cases of cyber-disruption at a seaport that widely depends on IoT technology are widespread. Another example is the port of San Diego, which had its entire network compromised by a ransomware attack and had to resort to several entities outside of its domain to ensure minimum functionality. This also highlights not only the necessity to prevent incoming attacks but also to ensure that they are detected early and, if already installed within the seaport network, have redundancy so that operations can proceed with minimal impact on all the inherent systems.

4.2 Services and tasks associated with Seaports

Associated with the increased technology usage throughout the world, seaports are no exception. IoT is, naturally, part of that growth process that is currently being observed [Agatić and Kolanović, 2020]. Some of the most prominent applications of this technology are:

- **Infrastructure:** Technology is an extremely important asset in this section, as it can allow for coordination efforts of large mechanisms to optimize its operations, as well as monitoring any signs of fatigue, wear, or cracks via sensors that can properly relay that information to a management center or outside devices. The nature of these sensors would be more resource-constrained since they should be placed throughout the entire seaport on a multitude of structures, so smaller more compact devices are preferred to optimize resources.
- **Cargo Management:** IoT technology also allows for cargo operation, whether via the usage of cranes or its travel via cargo vehicles, thus optimizing the entire seaport network. If automation is the goal, the resources used would need to be larger and quicker, so any usage of lightweight technology would not be advised, although it can still be used to guide vehicles via RFID sensors throughout their correct routes.
- **Customs' Process:** This part of a seaport's operations is more connected with backend systems that manage the traffic, companies, payments, and any cargo declarations. The goal here is to streamline these operations with more computational resources and increased redundancy to withstand any attacks or malfunctions. Again, there is no specific need in this area for lightweight IoT technology.
- **Security:** An essential asset to any seaport, proper security systems with surveillance and detection capabilities, powered by cameras, to detect and deter any physical attacks against the harbor. Since video streaming requires a high throughput and handles large amounts of data per second, lightweight devices are not fit to perform this task. In the specific case of encryption, lightweight versions can be used to speed up the process of protecting the transfer of information but not of resource-constrained devices.
- **Traffic management:** Also correlated with IoT technology, this area can improve how vessels and adjacent vehicles move throughout the seaport to optimize their routes and save crucial time which yields a higher margin of profitability. When it comes to the specific technology used, it would obviously depend on a variety of different sensors responsible for transmitting data for the weather, sea status, proximity, and route definition, among others. Also important to mention, similarly to the structure's management, some of these sensors and inherent devices to capture and transmit data (Arduino, Raspberry Pi, etc.) are battery-powered given their remote location, either at sea or in areas with difficult access, so it is important to

manage their power consumption to extend its usable energy for as long as possible and reduce any downtime period. Also associated with this is the necessity to encrypt the data is also imperative, so lightweight ciphers are preferable for this task.

- **Energy and environment:** Beyond the mentioned areas, another important area in which technology can aid is understanding where to save resources, reduce energy consumption, and thus dampen the environmental impact associated with seaport operations. Correlated with this is the usage of low-power-consuming devices, which run the minimal amount of tasks necessary. When it comes to security, lightweight encryption can also aid in providing adequate data protection at reduced power usage.

With this knowledge in mind, the following requirements can be outlined for this particular use case: data sent is usually small, ranging from a single bit to a small packet of information that only contains an integer or float, carrying out up to 64 Bytes of data. The speed of the data transmitted is crucial when automating activities at the seaport, which is why fast ciphers might be better suited for these applications. Only scenarios like Paper 4 referenced in Section 3.2, which do not depend on fast data access and can tolerate some delay will allow slower ciphers to be chosen. As for power consumption, any processes or transmission standards must not surpass a considerable level of power consumption given that most of the devices used might depend on expendable batteries and are at difficult positions of reach, like at sea in Paper 1, or integrated within structures to monitor fatigue and distance (Paper 3), both also analyzed in Section 3.2.

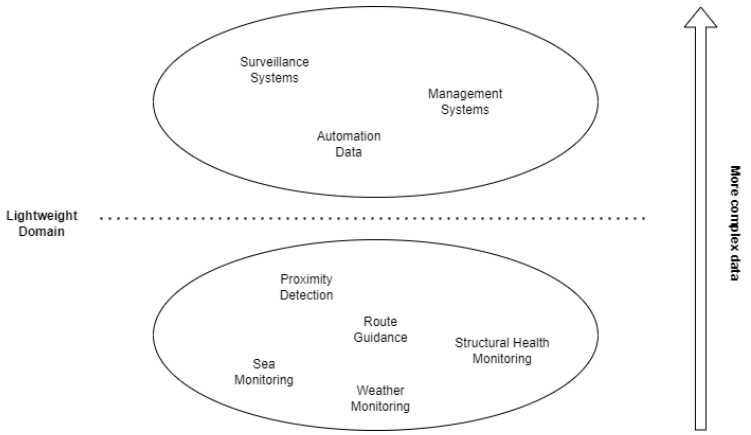


Figure 4.1: Services used in Seaport and their respective data complexity.

Chapter 5

Methodology and plan of action

The fifth chapter of this thesis will be of extreme importance as it will dive into the process and all the necessary tools and methodology to achieve the goals set out by this document. In Section 5.1 an overview of the available software implementations of the proposed algorithms is included to point out specifically which ones will be the target of further analysis. In Section 5.2 a comprehensive study of the algorithms previously selected is conducted to discover their known vulnerabilities, if any, as well as compare their security and, to some extent, performance. In Section 5.3 we overview and decide, based on the goals established and previously analyzed works, which metrics should be chosen to conduct the future analysis of the selected algorithms. In Section 5.4 we go over the scenarios that will be integrated in the analysis and comparison steps, providing details on how the data will be encrypted and which standards will be used to test the data transmission on each device. Finally, in Section 5.5 a concise plan of action is drawn, using all the information gathered in the previous sections, to define the workflow of the remainder of the thesis and of all the work done thus far.

5.1 Available Cipher Implementations

With the previous study into the used IoT devices in the use case of seaport, we concluded that both Arduino and Raspberry Pi are suitable candidates to advance the study. The Arduino model available for this study is an Arduino Uno, a very lightweight device that fits the context of resource-constrained which is crucial for this thesis. Furthermore, ESP32 and ESP8266 will also join the list of IoT devices given that they are also resource-constrained and can be useful comparison points to Arduino. There will also be three models of Raspberry Pi available for this study, namely, Raspberry Pi Pico, a lower-end, resource-constrained hardware that is comparable to Arduino Uno, and two other high-end devices that operate at higher levels in the IoT structure, handling larger volumes at data with a much faster processing speed, namely Raspberry Pi Zero 2W and Raspberry Pi 3B+. The following list contains the criteria for considering an implementation for each device as valid.

- **Arduino IDE:** Available libraries for the Arduino IDE or full code in C++. We tried to explore how easily code from other languages could be executed in Arduino but since its IDE only converts C++ language into machine-level code it would be considerably hard to use other programming languages. This item also includes the implementations for the Raspberry Pi Pico, ESP32, and ESP8266, since these three devices will also run on the IDE with the implementations found.
- **Raspberry Pi:** Since this device uses a Linux OS, we either selected downloaders for the encryption algorithm compatible with the Linux system or available code for encryption and decryption of the data.

The following table 5.1 details the available implementations of several lightweight algorithms for both devices, along with their reference. If the cell has a cross, no implementation was found in the public domain.

Cipher	Arduino IDE	Raspberry Pi
AES	[Sychra, 2023] [Weatherley, 2023a]	[Packetizer]
ASCON	[Weatherley, 2023b] [Weatherley, 2023a]	[Weatherley, 2023b]
CLEFIA	×	[Scarpa, 2016]
DES-L	×	×
DES-XL	×	×
HIGHT	[Walton, 2023]	[Walton, 2023] [Bossuet, 2016]
ICEBERG	×	×
KTANTAN	×	[Bossuet, 2016]
Lblock	×	[Bossuet, 2016]
LED	×	[Bossuet, 2016]
mCrypton	×	[Harttung, 2017]
Modified PRESENT	×	×
Modified QARMA	×	×
PICCOLO	×	[Pokala, 2020] [Jovanovic, 2012]
PRESENT	×	[Bossuet, 2016]
PRINCE	×	[Harttung, 2021]
QARMA	×	[Xu, 2022b]
RECTANGLE	×	×
Sato	×	×
SEA	×	×
SFN	×	×
SIMON	×	[McCoy, 2018]
SLIM	[Sen, 2022]	[Sen, 2022]
TEA	[Denhart, 2010]	[Xu, 2022a]
XTEA	[Protasowicki, 2021]	[Xu, 2022a]

Table 5.1: Implementation availability for the different algorithms.

5.2 Study of Algorithms' Security and Performance

The objective now is to evaluate their security and resistance to specific types of attacks to pre-determine which ones might be more valuable. it is important to note that this work is only theoretical and based on existing papers, with practical evaluations of interest to be conducted later on.

Throughout the next subtopics, there will be some mentions of attacks that can weaken or even compromise the security of each block cipher. The objective here is to give some context into each one of them and explain how they work to further contextualize and allow for a fair comparison of each algorithm's security.

- **Differential Cryptanalysis:** Technique used to analyze and break encryption ciphers by mapping out differences between pairs of plaintexts and their ciphers to discover any potential flaws that can weaken the algorithm. It is regarded as a powerful tool of attack and it has several subcategories, with the goal of gaining information about an encryption key.
- **Linear Cryptanalysis:** Usually more complex than differential cryptanalysis, focus on linear equations and statistical analysis to find common points between a plaintext, encryption key, and ciphertext.
- **MiTM (Meet-in-the-middle) / Biclique attack:** his one is a plaintext attack that targets ciphers that perform multiple encryptions in sequence. It works by storing intermediate values from an encryption operation that can improve the brute force attack time to obtain an encryption key. The biclique attack is a more specific version of the meet-in-the-middle tree, which relies on a biclique structure (complete bipartite graph) to gather information in between the encryption process. These attacks are famous for rendering the Double-DES not safe for usage and for discovering a brute force option for the AES cipher faster than the normal brute force time, attacking all rounds of the algorithm.
- **Related Key attack:** A very common type of attack that targets the possible relationship between two or more keys, assuming a connection between them that can weaken the security of the algorithm.
- **Side-channel attack:** Instead of focusing on the properties used within the encryption cipher, it targets any information disclosure that might be leaked upon its execution. These can vary from power usage analysis to variations in timing that can compromise the full integrity of the algorithm.

5.2.1 Individual Algorithm Analysis

AES

Starting with AES, this cipher has been widely known and used since 1998, being subject to numerous and rigorous applications throughout the years. To pinpoint more accurately its security, three criteria shall be used:

- **Time Security:** Determines the strength of the cryptography cipher against brute force attacks.
- **Avalanche Effect:** Determine whether small changes to the input greatly affect the corresponding encrypted output.

- **Strict Avalanche Criterion:** The Strict Avalanche Criterion (SAC) is satisfied if, when a single input bit is flipped or complemented, it causes every output bit to change with a 50% probability. In other words, even a small change in the input should lead to significant changes in the output, making it difficult for attackers to predict or analyze the behavior of the encryption function.

Another core concept is the difference between the Avalanche Effect and the Strict Avalanche Criterion. Whilst they both represent similar concepts, the avalanche effect is much broader and just highlights the necessity of the diffusion and confusion paradigm in order to drastically affect the encrypted text when any changes occur to the input, whilst the Strict Avalanche Criterion is a much more concise term, with a mathematical definition that states the necessary threshold needed to achieve validation from this criteria, providing a rigorous and quantifiable measure to measure the avalanche effect of a cryptographic cipher.

As it is known, AES has 3 available key-size options, 128, 192, and 256 bit size keys. The bigger the key size, the more protection is offered, although it comes with a drop in performance. Ideally, since the use case of this thesis concerns resource-limited devices, a smaller key size would be preferred. NIST recommends at least 112 bits of key size until 2030 and 128 bits of key size afterward, so theoretically the 128 option would respect the time security criteria. It also works extremely well in both hardware and software [Al-Mamun et al., 2017]. Table 5.2 represents the values in the studied paper for time to break each variation of the AES cipher. The time to break is given in years on the currently fastest available computer and so there is no incentive to choose any different option of AES to prevent brute force attacks.

Encryption Cipher	**Time to Break**
AES-128	3.19×10^{14} years
AES-192	5.88×10^{33} years
AES-256	1.0844×10^{53} years

Table 5.2: Time required to brute force different AES versions, in years. Adapted from [Al-Mamun et al., 2017].

Next, the avalanche effect was measured for the three different key sizes. To proceed, the test was applied to 8112 sample inputs with very small changes to either them or very small changes to the key used. The results are expressed in Table 5.3, where it is possible to assess that with an increasing key size, there is some noticeable increase in the desired avalanche effect.

Encryption Cipher	Average Avalanche Effect (%)
AES-128	49.977657
AES-192	50.036308
AES-256	50.042664

Table 5.3: Average Avalanche Effect, expressed in percentage. Adapted from [Al-Mamun et al., 2017].

Finally, the SAC analysis is also derived by the authors with a noticeable increase in the percentage value proportionally to the increased key size. In Table 5.4 there are two fields: average value which is the number of tests that passed the criterion calculus divided by 5, since for each cipher, five experiments were conducted, and percentage value, which expresses the previous value within the context of the 8112 sample inputs that were used to determine this criterion.

Encryption Cipher	SAC Value	
	Average Value	Percentage Value (%)
AES-128	4322	53.279
AES-192	4234.6	53.311
AES-256	4377.2	53.96

Table 5.4: Strict Avalanche Effect for three AES variants, expressed in percentage and average value. Adapted from [Al-Mamun et al., 2017].

With these calculations made, we believe that AES-128 would be the only option to be considered for the use case. Any difference in the criteria results is not substantial given the fact that the 128 key sizes, which would be much more efficient than their counterparts for resource-limited devices, are considered safe for the foreseeable future are the main reasons for this decision. It is also noteworthy to mention that the authors of this paper which analyzed the security of the AES cipher also proposed an improved AES method which, although yielded better results across all three criteria, we do not believe it should be pursued in this context, especially because the traditional AES has been around for years and been subject to extensive testing and usage, which give increased validity to its security.

Now, AES has been the standard for cryptography given that it is relatively fast and extremely secure, able to withstand as a safe cipher for several years. From further research, we found that there is an attack, called biclique attack, which is a type of meet-in-the-middle attack that was able to attack the full number of rounds of AES with a computational complexity of $2^{126.1}$ for AES-128, which is less than its 2^{128} complexity. However, this attack still leaves a very high complexity, which is only marginally lesser than the brute force option and thus cannot effectively compromise the security of the cipher.

ASCON

ASCON is another lightweight block cipher, which was released in 2014. As part of a lengthy process by the CAESAR Competition, this algorithm underwent rig-

orous security analysis. Without counting any error in its implementation, the best-known attack that can recover the key has a computational complexity of 2^{104} if the initialization is reduced to 7 from its 12 rounds, which is known as a Cube-like key-recovery attack, leaving a 5-round margin. Furthermore, excluding a brute force attack which has a complexity of 2^{128}, the best-known full-round attack is equivalent to 2^{130} and targets its permutation, reinforcing the initial trust in the cipher [Dobraunig et al., 2021].

Extensive analysis of its permutations was concluded, as well as tested against the most common and strong cryptography attacks and, so far, no important vulnerabilities have been detected, and, counting with a proper implementation and initialization, it is considered secure. Although its results are promising, it is still a relatively new cipher, and, even though it has surpassed most cryptanalytical attacks, it still doesn't have the same history and time used as, for example, AES, which in this field can matter.

HIGHT

HIGHT is a 2006 encryption algorithm based on block cipher that intends to enter the lightweight algorithms that still offer a good amount of security. Based on a 64-bit block length and 128-bit key size it aims to be used within constrained devices like sensors or small controllers [Hong et al., 2006]. To proceed with the analysis we first need to understand two concepts: confusion and diffusion. Whilst confusion asserts that the encrypted text does not possess any information or statistics about the plain text and thus both cannot be associated with each other, the diffusion paradigm asserts that a single bit change should yield a significant modification of the output (this concept is used for the avalanche effect described in the AES subtopic). According to a paper that compared HIGHT with the similar ciphers TEA, KATAN, and KLEIN [Bhardwaj et al., 2017], the authors do both a performance and security evaluation of the cipher.

To briefly mention the performance, HIGHT scored 2nd in lowest memory usage and 3rd in lowest power usage, losing to TEA in the last remark. Security-wise, the authors performed an analysis based on both confusion and diffusion. Starting with confusion, they created a 64-bit plaintext and a key with which they derived multiple similar keys with one-bit differences and encrypted multiple identical plaintexts, XORing the results. The percentage of 1s was calculated, yielding 49.21% for the HIGHT cipher, mildly above TEA by just 0.07%.

For the diffusion paradigm, a similar study to AES was conducted to determine the avalanche effect, which yielded 49.7% diffusion for HIGHT, against 51.1% for TEA. A difference of 1.4% is significant but not definitive in excluding this algorithm.

Furthermore, from the research conducted HIGHT does not appear to be widely used, not as much as the TEA/XTEA or AES ciphers, which in the field of cryptography is not desired since security usually comes from intense scrutineering and public use. Once again, this cipher is heavily dependent on proper implementation, as it can open the door to some attacks like, for example, side-channel

attacks, although the authors performed a comprehensive security analysis of their algorithm, implementation, and S-Boxes to ensure they are safe against a multitude of attacks. It can also, theoretically, suffer from the same type of attack as AES, biclique cryptanalysis to weaken its key structure, again yielding a new computational complexity of 2^{126}, identical to AES.

SIMON

Although SIMON is not available for Arduino IDE, we still decided to conduct some analysis on it since one of the papers read really highlighted its security and unparalleled efficiency. In terms of vulnerabilities, no known attack can compromise the entirety of rounds of the SIMON cipher; although some approaches suggest that using differential attacks can attack some rounds of the algorithm. Another study [Chen and Wang, 2016] used linear hull attacks to target more than half of the rounds, leaving a margin of safety of 12 or 13 depending on the SIMON cipher used (96 key size or 128 key size).

Performance is where SIMON excels. The area occupied is significantly smaller than both XTEA and AES, achieving very low power usage which can be of great value for power-restricted devices. It is also excellent to be used in hardware and given the fact that it supports multiple key sizes, its performance and power-saving can be adjusted given the task and secrecy of the task in hand [Adriaanse et al., 2021]. Furthermore, as with most ciphers, these statistics plus memory usage will depend on the implementations; whilst some focus more on saving power and resources, others aim at providing a higher throughput, at a cost of higher power consumption. What stands out with this cipher is its ability to modulate and use different key sizes to balance security and performance.

TEA

The TEA cipher is not considered safe, due to two reasons. First, and more grave, due to the nature of the rounding function, each key has 3 equivalent keys that will yield the same encryption and decryption. Secondly, it can also be subject to some related-key attacks, that can theoretically occur, although very impractical [Adriaanse et al., 2021]. Furthermore, the security analysis for the confusion-diffusion paradigm appears to be similar to another lightweight cipher, HIGHT, which was previously discussed in the HIGHT subtopic.

Since we're dealing with resource-limited devices, their performance is also important. It appears that this cipher requires low energy usage but higher memory usage, at least comparatively to HIGHT [Bhardwaj et al., 2017].

XTEA

XTEA was built as a successor to TEA, with the main goal of addressing the security concerns that made TEA non-viable for high-security environments. However, when the number of rounds is reduced, some studies point out less than

36 rounds, and some related-key attacks or meet-in-the-middle attacks can arise [Adriaanse et al., 2021]. The main point of XTEA is to reduce the number of rounds used to increase performance but, at the same time, ensure the safety of the algorithm. Some studies, particularly [Khan and Moessner, 2011] point out that XTEA is significantly weaker than AES.

When performance is concerned, their results greatly depend on the implementation developed. The consensus is that XTEA provides a good speed combined with medium power usage which, although lower than AES, is still significantly higher than TEA. Other implementations aim at providing higher output at a cost of more power usage, which in our use case would probably not be recommended given the type of devices we're dealing with. It is also important to remember that the number of rounds used is directly associated with the performance branch, allowing for some optimization and efficiency gains.

5.2.2 Comparisons between algorithms

Throughout this chapter, several block ciphers were studied to infer their inherent security and to highlight potential performance problems. Six algorithms were delved deep into, five of them with available implementations for both Arduino and Raspberry Pi, while the other one (SIMON), although only available for the Raspberry Pi, was highlighted as a safe and extremely lightweight cipher throughout the research.

From a security point-of-view, TEA shows some concerns since it possesses some vulnerabilities that can significantly weaken and compromise the security of the protocol. From the remaining five, AES is the most tested and attack proof currently known, being subject to countless avenues of attack and withstanding all of them, the exception being with the biclique attack which slightly weakened the cipher, but not to a point of any concern. HIGHT and XTEA scored similar values of diffusion and confusion and are closely followed by SIMON so any of them would also be a good choice. ASCON is the upcoming lightweight cipher that, so far, was able to guarantee security with the only attacks able to compromise portions of the key being to a maximum of 7 rounds, all within the security margin.

Although this will be analyzed in much greater detail in future topics, efficiency, and power saving are crucial aspects of any lightweight encryption algorithm. In this regard, ASCON and SIMON are at the top of the list, with unparalleled power saving and efficiency. HIGHT, XTEA, and TEA are then closely matched as second best, with XTEA slightly behind given the fact that it occupies a larger area and requires more power to efficiently function. Although AES can be tuned to use smaller key sizes which are still considered safe, its energy consumption appears to be larger than the other ciphers, which is to be expected given the security that it demonstrates.

Cipher	Source	Area (GE)	Power (W)	Throughput (kbps)
XTEA	Kaps (2008)	3490	19.5	57.1
XTEA	Kitsos et al (2012)	3490	61	200
SIMON	Yang et al (2015)	944	0.762	4.2
SIMON	Yang et al (2015)	1403	1.239	133.3

Table 5.5: Different available implementations for some ciphers. Adapted from [Adriaanse et al., 2021].

In the previous Table 5.5, we have a brief comparison of two versions of XTEA (the first one is more power-saving than the second one), compared with two other versions of the cipher Simon. From the numbers, we can quickly realize that XTEA requires a much larger implementation area and power to function properly which is a huge downside of it. From the analysis conducted thus far, we can justify this discrepancy largely on both the age of XTEA and the solutions implemented to mitigate the problems of its predecessor, which had an impact on its performance. It is also important to mention however that this data was gathered using implementations described by different papers. The only downside that SIMON has is that, as expressed in Table 5.1, no known implementations for Arduino have been made, which significantly compromises its analysis in any future chapters.

Cipher	Key Size	Best Key Recovery Time	Known Serious Vulnerabilities	Date Published	Energy Consumption
AES-128	128	$2^{126.1}$	×	2001	16.7
AES-192	192	$2^{189.7}$	×	2001	-
AES-256	256	$2^{254.4}$	×	2001	-
ASCON	128	2^{128}	×	2023	-
HIGHT	128	$2^{126.1}$	×	2006	25.5
SIMON	96 or 128	$2^{83.74}$ or $2^{120.47}$	×	2013	2.3
TEA	128	2^{32}	✓	1994	30.3
XTEA	128	$2^{120.65}$	×	1997	70

Table 5.6: Characteristics of compared algorithms.

Cipher	Differential Cryptanalysis -	Linear Cryptanalysis	MITM/Biclique	Related Key Attack	Side-Channel Attacks
AES-128	✓	×	✓	✓	✓
AES-192	✓	×	✓	✓	✓
ASCON	-	✓	-	-	-
HIGHT	✓	✓	✓	✓	×
SIMON	✓	×	×	✓	×
TEA	×	×	×	✓	×
XTEA	×	×	×	✓	×

Table 5.7: Attack susceptibility of the algorithms.

Now, discussing the previous tables 5.6 and 5.7, these summarize what was said throughout this topic, to help us understand which ones to follow or discard from the in-depth analysis. All of the highlighted ciphers have reasonable key sizes available and most of them are proofed against practical attack demonstrations, the exception being the TEA algorithm which, due to the reasons highlighted previously, is not considered safe, thus originating the XTEA variant. Another important factor is the date published, as older algorithms are more secure-proofed

given the fact that they were extensively analyzed and used for longer periods of time. Although not extremely relevant from a security point of view, their energy consumption also carries some weight when doing this comparison, with SIMON delivering the best results in theory by far. As discussed previously, XTEA appears to demand more resources, compromising its choice as a very lightweight cipher (this could also be attributed to the fact that this cipher, as well as TEA, are very old with more than 25 years since they were first published, which can somewhat contribute to the less optimal energy consumption). Beware that Table 5.6 mentions software implementations under the same device, which unfortunately does not provide data for ASCON and 192 and 256 AES variants.

One last important step is whether or not these ciphers are standardized, as this carries some weight in recognizing the importance and security of an encryption algorithm. As showcased in Table 5.8 AES and SIMON are both standards, although SIMON is only a standard for RFID lightweight encryption after some discussions about potential weaknesses, lack of need for a new encryption method for lightweight devices, and an existing backdoor by the NSA (which was present in their previous Dual_EC_DRBG cipher). ASCON is on the path to becoming a standard after winning the CAESAR competition for lightweight ciphers but, since it was published in 2023, standardization could still be a few years away if it indeed occurs. Both TEA and XTEA have applications in their fields of encryption but given their age it is not expected for any of them to be standardized.

Cipher	Standardization
AES	Standard encryption method for symmetric encryption and widely used, standardized in ISO/IEC 18033-3 and FIPS PUB 197 (NIST publication document)
ASCON	Won the CAESAR competition by NIST for lightweight and secure cryptography in early 2023 and is currently being prepared for standardization, currently associated with the document NIST IR 8454 that defines the process of evaluation and selection of the cipher.
HIGHT	Published in 2006 and used in some areas of lightweight encryption, standardized under the norm ISO 18033-3:2010, which defined, in 2010, trustworthy block ciphers.
SIMON	Failed to be published initially as a standard due to some weakness concerns, ended up being standardized by ISO under an RFID air interference document (ISO/29167-21)
TEA	Not a standard, with several security flaws
XTEA	Not a standard but has several applications in lightweight encryption.

Table 5.8: Standardization status of the algorithms.

5.3 Evaluation Metrics

The metrics definition is an essential step in programming how the study will be conducted. They will allow comparisons to be made between the different algorithms to determine their strengths and weaknesses and, ultimately, make any recommendations about their usage in the use case of seaport easier. This section is separated in four type of metrics: (1) metrics that concern the algorithm, (2) metrics related to the implementation code, (3) time and speed metrics, (4) and power metrics.

5.3.1 Algorithm Metrics

The first section of comparison metrics is specific to the algorithm used and its inherent characteristics such as: key size, block size, and number of rounds, as mentioned in Section 2.3. These will provide some context into each cipher's way of operating and should give an initial picture of how efficient and safe the algorithms might be, which should afterward be correlated with more data from more thorough metrics.

5.3.2 Memory Metrics

Another important metric is memory usage which can be measured in Arduino and ESP devices with the help of a module called MemoryFree [McNeight, 2016], or after compiling the code if there are not that many memory changes throughout the program's logic, whilst for the Raspberry Pi this value can be measured with the command "grep MemTotal /proc/meminfo". This, however, will tell the RAM used by the full system, which can vary greatly with time and thus not yield correct results, with which good conclusions can be drawn. A possible solution to this problem could be the usage of a tool called Valgrind, which is available for Linux, and can tell the memory usage of a single executable/program, allowing for a much more straightforward comparison ("valgrind –tool=massif <executable> <arguments>") [El-hajj et al., 2023].

5.3.3 Time and Speed Metrics

These metrics are relevant because they solely rely on how each cipher performs on each device and give a more practical and comparable outlook on their efficiency and speed. They were adapted from [El-hajj et al., 2023].

1. **Time**

   Starting with the time it takes to process each stage, this will be critical to measure how efficient an algorithm is and valuable information for those scenarios in which fast ciphers are preferred. To apply this to our testing routine, we simply need to measure the time immediately at the beginning of the process to be evaluated and at its end, thus calculating the delta between these two.

$$\Delta t = t_e - t_s \tag{5.1}$$

   (a) Δt is the time measurement from the beginning to the end of the process in µs.

   (b) t_e is the time at the end of the process in µs.

   (c) t_s is the time at the beginning of the process in µs.

2. **Speed Throughput**

   The second unit with which we can compare the algorithms is speed throughput. It is measured in bytes per second, and it aims to understand how fast each process of encryption and decryption is for each algorithm.

$$s_t = \frac{S}{\Delta t} \tag{5.2}$$

   (a) s_t is the speed throughput in bytes per second.

   (b) S is the text size in bytes.

   (c) Δt is the time measurement from the beginning to the end of the process in seconds.

3. **Time per Byte**

   Next, we have time per byte, which represents the amount of time it takes to encrypt/decrypt one single byte of information, to serve as an auxiliary measurement to both speed throughput and time consumed.

$$s_b = \frac{\Delta t}{S} \tag{5.3}$$

   (a) s_b is the measurement of time per byte, expressed in µs/Byte.

   (b) Δt is the time measurement from the beginning to the end of the process in µs.

   (c) S is the text size in bytes.

4. **Speed Latency**

   Next, the speed latency is a measure that takes into account the frequency of the processor in Hz and the block size of the algorithm in Bytes to give a more fair comparison ground for different algorithms (thus using block size as a metric) and different testing platforms (thus using frequency as a metric). The resultant value of the multiplication between these two previous values is then divided by the speed throughput to yield a speed metric that can be used to establish fair comparisons.

$$s_l = \frac{f \times B \times \Delta t}{S} \tag{5.4}$$

   (a) s_l is the speed latency in cycles per block.

   (b) f is the processor's frequency in Hz.

   (c) B is the block size used by the algorithm in bytes.

   (d) Δt is the time measurement from the beginning to the end of the process in seconds.

   (e) S is the text size in bytes.

5.3.4 Power Metrics

Also important to our study are the power-related metrics, indispensable to assess how each cipher performs on devices that are resource-constrained and thus might need power-efficient solutions. The two proposed metrics in this subchapter were again adapted from [El-hajj et al., 2023], summarized in Section 3.4.

1. **Power Measurement**

   This is the most simple value to obtain, and it aims at providing a raw output of the resultant power measurement tool to express the power amount that is being used by the IoT device throughout the encryption/decryption process. The measure used is Watts, which is equivalent to a Joule per second, and uses both the volts and amperes measured by the multimeter (or other similar device) to calculate the power.

   $$P = A \times V \tag{5.5}$$

   (a) P is the power in watts.

   (b) A is the electric current in amperes.

   (c) V is the electric potential in volts.

2. **Power Latency Measurement**

   Finally, one other potentially interesting metric is power latency which provides the power usage per bit of encryption or decryption, to help understand how efficient (or not) each cipher is.

   $$E = \frac{P \times \Delta t}{S \times 8} \tag{5.6}$$

   (a) E is the power latency in µJ per bit.

   (b) P is the power in watts.

   (c) Δt is the time measurement from the beginning to the end of the process in µs.

   (d) S is the text size in bytes.

5.4 Data Usage Scenarios

This chapter is subdivided into three steps, one that defines how the study will be conducted when testing each algorithm's performance and security in both selected devices, and three more that summarize the requirements needed to test the data transmission with different standards, Wi-Fi, Bluetooth, and ESP-Now, and thus get comparable data between their performance upon sending the encrypted information.

5.4.1 Testing encryption on the IoT device

The first test, which will be the central part of this thesis, will focus on analyzing and comparing different lightweight encryption ciphers on different IoT devices, namely Arduino Uno, Raspberry Pi Pico, 3B+, Zero, and both ESP32 and ESP8266. For that, we will apply the previous metrics and determine the pros and cons of each algorithm according to their performance based on those metrics.

In this test, the raw data will be emulated in the code running the encryption ciphers, mimicking the type of data that a sensor or multiple sensors would output. Any ciphertext produced will remain in the device, with performance and efficiency metrics being measured throughout the stages of encryption and decryption of the content.

5.4.2 Transmission via Wi-Fi

The second testing campaign relates to data transmission over Wi-Fi. This is done to both assess the security of the cipher and the efficiency of the transmission protocol when transmitting the data.

Several devices at our disposal can use Wi-Fi as a transmission standard however, for this study, we will consider both ESP8266 and ESP32 to understand how each device's performance varies while being used as an Access Point or simply a Wi-Fi client and how these two modes compare in terms of power spending against the other used standards. With the installation of the packages for both ESP devices, some built-in classes and examples instantly support several capabilities of Wi-Fi, such as a scanner to assess the network performance in terms of efficiency, strength, and maximum operational range, and client/access point functionalities useful to test these different roles between the two ESPs and determine how they affect their consumption and if this value has any variability by simply changing the device's order.

5.4.3 Transmission via Bluetooth

Another key aspect of the testing campaign involves Bluetooth, which we will take advantage of both its classic and low energy BLE distributions. Their main cases of application will involve short-ranged transmissions which demand a need for low power usage, thanks to the low energy module available.

In order to apply this concept, we'll have to separate its study into two parts, one for the ESP32 and another for the Raspberry Pi Zero and 3B+. The former will use Arduino IDE and be built around the examples available with the installation of its software under the Bluetooth tab to understand how the protocol works and mold it to fit into the research program, more specifically, how to modify the transmission rate and packet contents so that we can accurately portrait the necessary small packet sizes that fit into the criteria proposed in the state-of-the-art. As for the Raspberry Pi devices, since the Arduino IDE is not an option, we

will use Python with the serial library to establish a connection with a remote device and run all the needed tests.

5.4.4 Transmission via ESP-Now

Lastly, one other important communication protocol is ESP-Now. Its usage is designed for IoT environments with ESP hardware, given that it uses less bandwidth, has a much larger range than both its Wi-Fi and Bluetooth counterparts, and draws a compared amount of power when compared to other protocols apart from BLE. Given these characteristics it could be applied in the seaport use case, especially to cover those scenarios in which data needs to be carried by considerable distances.

How could this standard be applied to our study? Well, it would only have practical applications for ESP devices, however, one could use the Arduino IDE since both hardware support their usage to receive, compile, and run the code provided by it. The purpose of ESP-Now within the parameters of the project would be to establish a connection and exchange data between an ESP32 and an ESP8266, which is crucial to demonstrate the capabilities of this standard and assess how well it performs compared to other transmission mechanisms. The code and subsequent documentation to understand and fine-tune ESP-Now is available in the documentation for the standard [Espressif Systems, 2024].

5.5 Plan of Action

In this section, we compile all the findings gathered in the methodology to propose a plan of action on how to build our research for the following chapters. Starting with the algorithms that are going to be part of the comparison, AES, ASCON, TEA, and XTEA were selected. This decision was based on a variety of criteria and each has an individual reason to be part of the study moving forward. AES is the standard of lightweight encryption, has been subject to extensive cryptanalysis throughout the years, and has a variety of different trustworthy implementations, far more than any other cipher. We intend to conduct the tests on all three versions of AES with different key sizes, namely 128, 192, and 256 bits of length to also understand how performance can be affected by the increased key size. ASCON was chosen because it is a very recent algorithm that won the CAESAR competition; establishing direct analysis between ASCON and the other ciphers will also meet one of the goals of this thesis, understanding possible advantages and disadvantages of newer lightweight ciphers. Finally, TEA and XTEA were also chosen as the last algorithms given their publicly available implementations for the devices at hand, as well as their very lightweight parameters, which might be useful for extremely constrained scenarios. It is noteworthy that in the intermediate plans of this thesis, HIGHT was the preferred choice over these two ciphers, however, given that later analysis found that the implementation did not compile and execute on the Arduino IDE, despite numerous attempts otherwise, the choice to change the algorithm was made.

Now that the four ciphers on which to conduct the future tests have been selected, we have to decide on the ground to compare them and objectively measure their performance. Starting with the metrics, all of the previously mentioned ones in Section 5.3 will be used as they cover all the necessary data needed from the performance of both the cipher and IoT device, to their security. Furthermore, we will use a USB Power Meter to gather all power-related metrics, and build-in code to get data, timing, and overall code statistics about each algorithm.

With these three objectives decided we just need to define the scenarios in which to conduct said tests. As was previously mentioned, our evaluation process will be done in two phases: an initial testing sequence where the chosen algorithms will be implemented in both devices, and a validation of its functionalities and performance will be done resorting to the selected metrics, with the goal of comparing each cipher and better understand their weaknesses and strengths, as well as analyzing each device's behavior to point out any differences in execution that might suggest that one is better than the other in the specified use case. After this, the second phase will be comprised of testing the transmission of the encrypted data using the three selected transmission protocols, namely Wi-Fi, Bluetooth, and ESP-Now, to understand their operating differences and their performance based on their speed and efficiency of transmission.

As expressed in Figure 5.1 we have the main tasks that were performed throughout the intermediate step of the dissertation, along with the time consumed by each one. The writing of the intermediate document was the most lengthy process given the amount of information gathered and reviewed, as well as defining a concrete methodology that can support the future tests and comparisons proposed. Work began in mid-September, with a meeting with both advisors outlining the initial structure, objectives to achieve, and early work such as reviewing papers and information that concern both IoT-driven low-power devices and available lightweight encryption algorithms. Every two weeks a meeting would take place to discuss the work done up until that point and agree on the next decisions and work to be carried out.

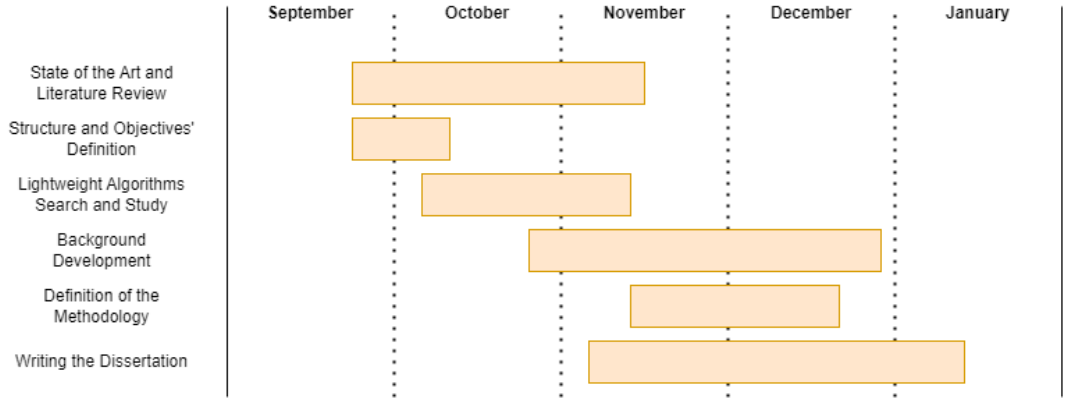


Figure 5.1: Intermediate tasks done and time consumed using Gantt diagram.

As for the second and final wave of tasks, Figure 5.2 provides a graphical depiction of them and the time frame required for each one. Starting the second semester, implementing all the ciphers on the devices was a top priority since

this would be the foundation of all further steps and analysis. This period was extended when compared to previous estimates due to the complexity of having an increasing number of devices and different implementations. Throughout this initial step, metrics were also collected and validated at the same time, to optimize the time spent on this portion of the thesis. Around mid-March, comparisons and the necessary charts started to be made, in conjunction with validating and assessing the three transmission standards in use for this work. It is also noteworthy to mention that in previous expectations, any tests of the transmission standards were supposed to be isolated from all previous tests, however, given the goals revised after spending more time than anticipated with the encryption process, this task had to be done in par with the previous one. Finally, around the start of April, the final thesis was started, and it reached its conclusion in the final days of June.

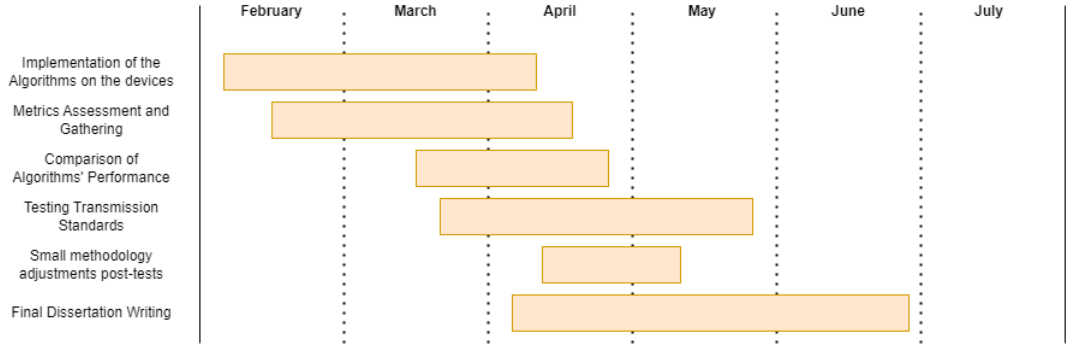


Figure 5.2: Final tasks done and time consumption using Gantt diagram.

Chapter 6

Analysis of Ciphers' and Devices' Performance

The sixth chapter of this thesis will be dedicated to the comparison and assessment of each cipher's and device's performance. To start, in Section 6.1 an initial overview of how each cipher and device were validated, to ensure that the necessary measurements can be accomplished to reach the desired goals of this study. Section 6.2 complements the previous section by providing further insight into what tests were conducted, with great detail, so that any procedure is properly documented. In Section 6.3, the results of all mentioned tests produced on the ciphers are presented in several figures, with each being analyzed to draw the necessary conclusions. This section is subdivided into three categories, one for the Arduino Uno, another for the ESP family of devices and another one for the three Raspberry Pi used throughout the study. Furthermore, each category contains information on their time and speed characteristics, as well as analysis conducted on their power consumption, memory used, different input sizes, and different modes of operation for some algorithms. In Section 6.4 the devices are then compared based on their previous results. Finally, in Section 6.5, some scenarios that are correlated with seaports are drawn out, so that all the knowledge gathered in this chapter can be applied to them.

6.1 Preliminary Installation Setup

To start the testing campaign, several steps must be undertaken to ensure the right setup is in place. Firstly, we must validate the available implementations and gather the required metrics on one device. Given its simplicity and previous knowledge of its methods of operation, the Arduino Uno was selected. The first run was used to assess whether the encrypted values were accurate with the chosen plaintext, a small payload ranging from 8 to 64 bytes depending on each cipher, as well as to confirm if the decryption results matched, once again, the original data. This first step was successful for all encryption algorithms, as was expected given the implementations' fidelity. Then, an initial data assessment using only limited time and speed metrics was carried out to get a first idea of the

capabilities of the Arduino Uno and to accurately portray each formula to yield the correct results, which was also successful.

Afterward, we applied the same data and metric validity to the ESP32, ESP8266, and Raspberry Pi Pico, as these also took advantage of the Arduino IDE and its libraries, thus facilitating the device transition. Beyond assuring that the returned values were correct, it was also part of this process to understand if the results were within the expected values, more precisely, if they matched with equal performance tests done on similar hardware, as well as if the time and speed distribution accurately portrayed the difference in technical capabilities and, more specifically, clock speed. These tests also yielded the expected outcome for all devices tested.

We also had to validate the power data and the device we would use to gather those values, a UM25C USB Power Meter. Again, given its simplicity, the Arduino Uno was chosen as the first device used to perform these tests, which consisted of installing the PC software for the logging capabilities, as well as understanding how to use it and gather the necessary data from it. Given that the device displays each power consumption measure every half a second, extracting all of the information to a separate Excel file was also necessary, so that we can match each timestamp with a particular operation. In Figure 6.1 we have the display for an Arduino Uno. The two most important parts of said software are: 1- Voltage, current, and power measurements which, as previously stated, are updated every half second; and 2- a graphical representation of both voltage and current in time, to better understand the power variations as the computational task changes. The changes from idle, encryption, and decryption were noticeable, and further validation was also extended to the remaining devices.

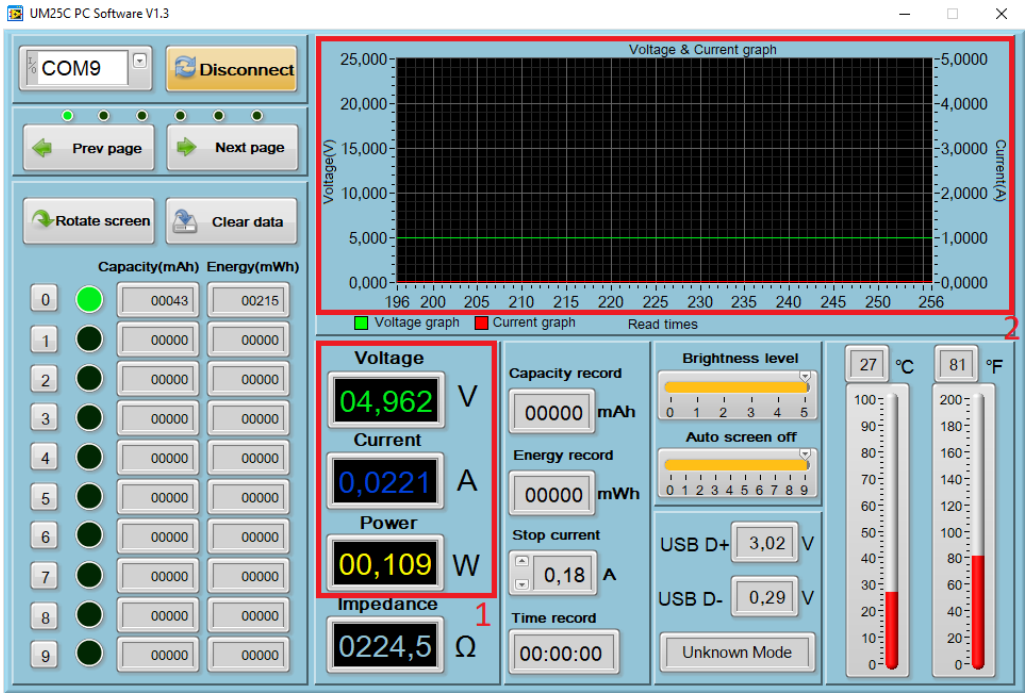


Figure 6.1: Screen capture of the UM25C PC software for logging data.

The last piece of validation needed was to extend the testing campaign to both the

Raspberry Pi 3B+ and the Zero W. Since it is not viable to run any code through them via the Arduino IDE, C++ versions, already tested by third parties on Raspberry Pi hardware, were used and compiled. Both encryption and decryption validation were performed once again and were successful. Beyond this, all that is left is to test some speed and time metrics and assess if the power data is also viable, which was also correct.

These tests took some time, but were fundamental in understanding how to operate and fine-tune each device to extract all the necessary data to have viable and important metrics to compare both the devices and the ciphers' performance.

6.2 Testing conditions

After the initial period in which all ciphers and devices were validated, a routine of tests had to be implemented that followed the guidelines mentioned in the methodology of this thesis, with each data analyzed having 16 Bytes of size. All devices had three phases of testing when it came to encryption and decryption:

- **Statistical data gathering:** This first stage would perform a round of encryption and decryption on the same block of data and repeat it for a certain number of times (dependent on each device, usually to increase confidence in the results, several runs would be conducted to determine the ideal number of runs that does not heavily stress the device and also provides accurate data), between 500 and 1000, measuring both the starting and ending times, to get the time it took to perform those operations. Afterward, all of the needed metrics concerning time and speed would be outputted and saved onto a file, using a serial monitor grabber code developed in Python.
- **Value variation:** In order to understand the variation of time between each singular operation, the previous test was revised, so that the time would be measured at the start and the end of one singular encryption/decryption, instead of measuring the entirety of the operations conducted. For all ciphers and devices, 500 operations (either encryption or decryption) were executed to determine the spread of the results. Using the same Python code, this data was saved onto files, highlighting all the values so that statistics like standard deviation and variation figures could be made.
- **Power measurements:** Lastly, the code from the first test was reutilized, without printing the statistical data, to measure the power consumption of several operations. By printing the timestamps on which each operation would start and end, it was easy to correlate these with the Excel data provided by the Power Meter.

Furthermore, we also performed tests to understand the memory usage for the Arduino, as well as testing different modes of operation of some ciphers for Arduino and ESP32. These two tests will provide further insight to complete the remaining objectives. In all Figures related to the display of results, the first column corresponds to encryption while the second represents decryption.

6.3 Algorithms' Analysis and Comparisons

This section provides the individual analysis results of each cipher's performance in terms of time, speed, power consumption, memory allocated, and, for the Arduino and ESP32, different modes of operation for some block ciphers. The study carried out assessed how each algorithm behaves per device and if their performance is consistent or not across different types of hardware.

6.3.1 Arduino Uno

This section will provide all of the data gathered about the Arduino, as well as a critical analysis of the performance of each cipher when compared to one another.

Speed and Power Outputs

As mentioned, the first tests were conducted on an Arduino Uno, given its simplicity and familiarity with operating it. Looking at Figure 6.2 it is possible to draw some initial conclusions. Firstly, as the key size increases in AES, a gradual increase is also noticeable in the time per encryption and decryption. Again, that same increase also occurs with TEA and XTEA, which makes sense given the fact that the latter is newer and implemented some security nuances that escalate their complexity. One other important factor is the difference between a single encryption on the different ciphers, decrypting with AES is, on average, 600 µs slower when compared to encrypting but this difference is drastically smaller for TEA and XTEA (only around 40 µs and the other way around, with encryption being slower) and almost nonexistent for ASCON.

When it comes to power consumption, Figure 6.3 gives a raw image of the power, in Watts, required to perform one encryption and decryption, on average. As we can see, the three variants of AES consume less power when compared to the remaining three ciphers, however, these differences are small, around 0.001 W when compared to XTEA and no more than 0.007 W when compared to ASCON. Figure 6.4 provides a better picture of the energy consumed by each cipher per encrypted and decrypted bit, taking into account the time it takes to perform one operation, on average, as displayed in the figure 6.2.

Figure 6.5 showcases the difference between power consumption while in idle (not performing any meaningful operation) and doing both encryption and decryption. Two important things to note, one is that the percentage difference is very small, with the highest margin being 3.17% using AES for encrypting, and the second one is that, for XTEA, and even TEA on a much smaller scale, it draws less power performing a single encryption and decryption when compared to the device being on idle; this occurs most likely given a combination of two factors: the resource-constrained nature of the device, in which all results do not diverge greatly from the average idle time, and the simplicity of both TEA and XTEA when compared to the other ciphers. Both of these findings once again corrobo-

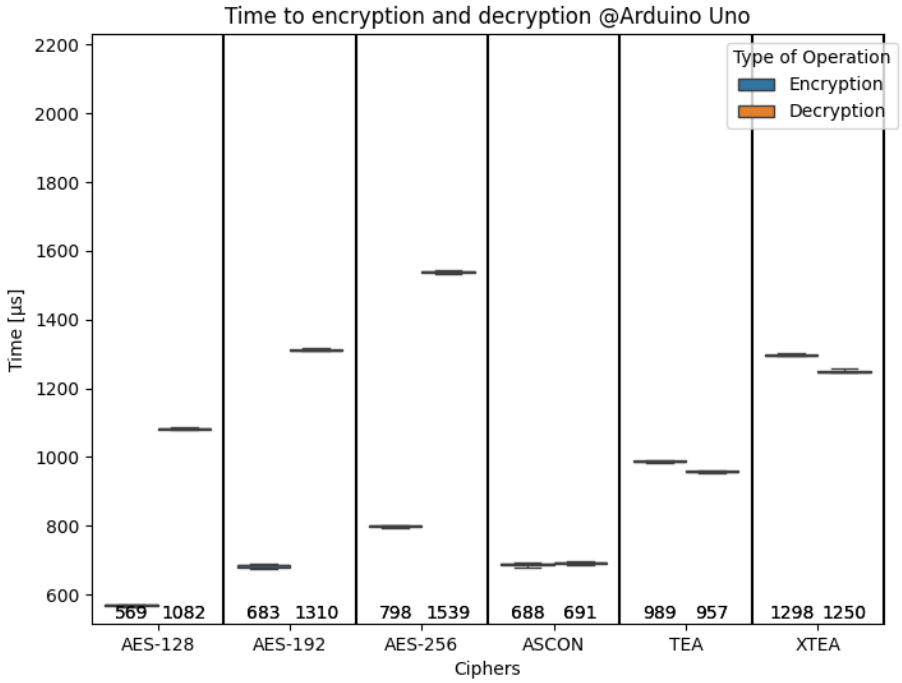


Figure 6.2: Time in µs for one singular encryption and decryption.

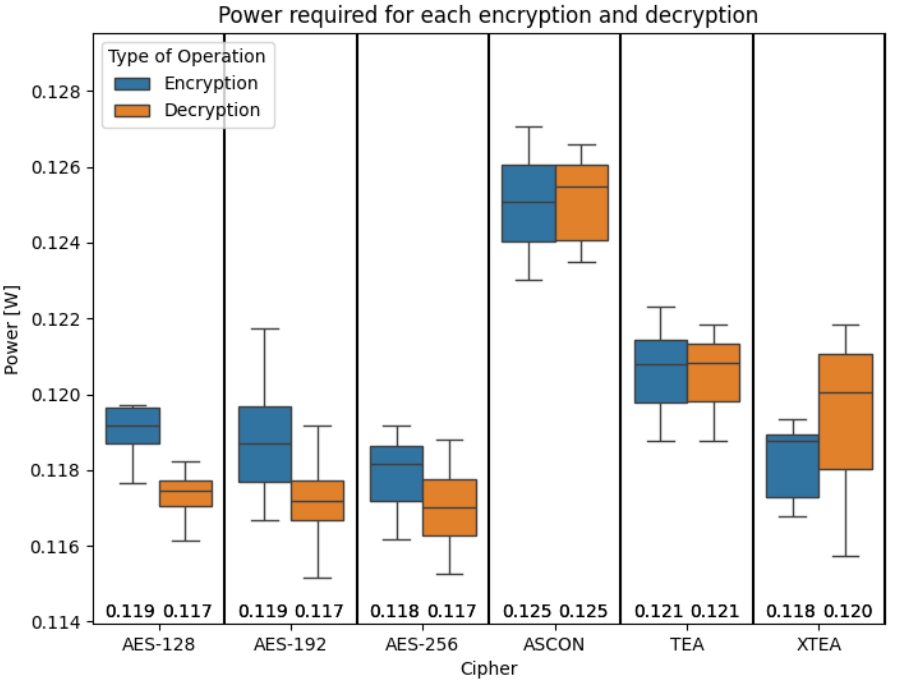


Figure 6.3: Power consumption of each cipher on an Arduino Uno.

rate the hypothesis that with more constrained devices, any variation in both the speed and power measured are considerably small, although further chapters of this thesis will provide further context with more complex devices.

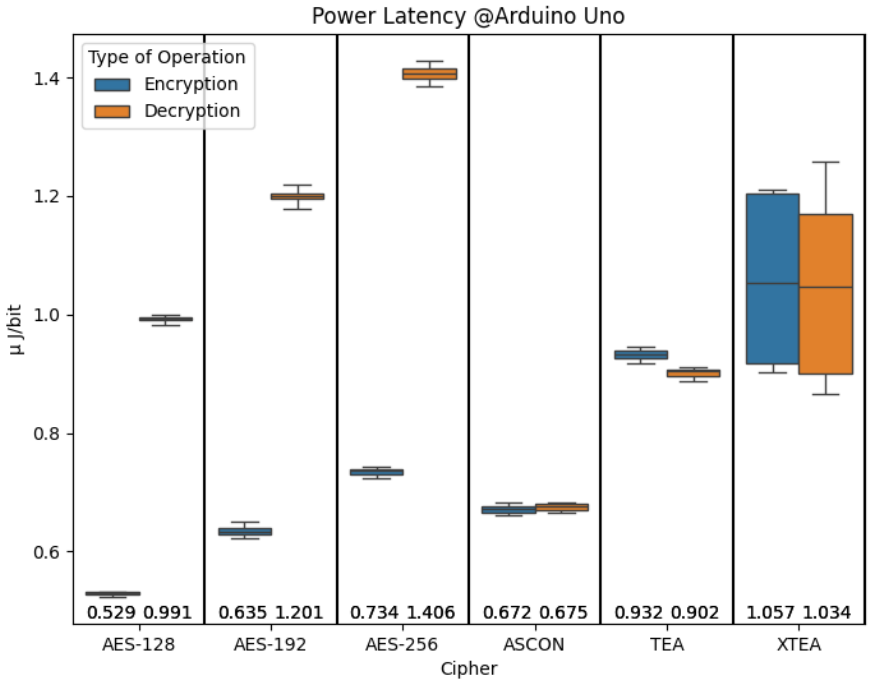


Figure 6.4: Power Latency on an Arduino Uno.

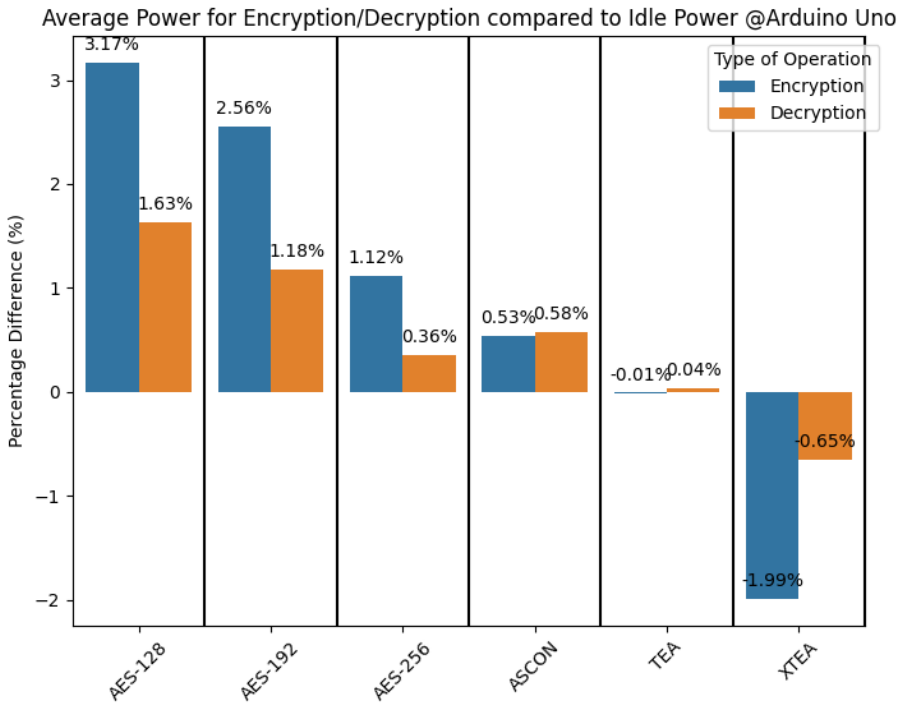


Figure 6.5: Power consumption difference between average encryption and decryption power compared to the average idle power, on an Arduino Uno.

Different Input sizes

Another important analysis point stems from different input sizes to understand how different payloads can impact performance, if at all. Since we are dealing with an Arduino Uno, any input needs to be relatively small, so as not to exceed any memory limitations of the device. Equation 6.1 takes into account the time it took to perform encryption and decryption, as well as the size of the buffer so that a fair comparison can be made between the different inputs.

$$\Delta t_{adjusted} = \Delta t / (Buffer/16) \tag{6.1}$$

1. $\Delta t_{adjusted}$ is the time measurement from the beginning to the end of the process in seconds, adjusted to compare to buffers with 16 Bytes.
2. Δt is the time measurement from the beginning to the end of the process in µs.
3. **Buffer** represents the total amount of data encrypted.

In Table 6.1 we have 4 key parameters, with the first three representing three different measurements of time which correspond to the average of 1000 runs. The first is a mere representation of the time it took to perform one encryption with each type of buffer as the input, the second named "Time Adjusted" takes advantage of the previous formula to get the necessary time to encrypt the 16 Bytes of data. Lastly the "Time for 16 Bytes" results from the modified payload for all ciphers only to process 16 Bytes at a time. The estimated and actual results for both ASCON and XTEA are quite similar, however, if we look for the TEA column, these two results are more than 100 µs apart, which is quite a bit of margin. Given the memory constraints of the Arduino Uno, no payloads bigger than 64 Bytes were tested, however, these results already show the discrepancy between some of the implementations.

Metrics	AES-128	AES-192	AES-256	ASCON	TEA	XTEA
Time (µs)	566	680	794	339	556	5192
Time Adjusted (µs)	566	680	794	679	1112	1298
Time for 16 Bytes (µs)	566	680	794	685	988	1316
Buffer (Bytes)	16	16	16	8	8	64

Table 6.1: Difference between average time and average time adjusted on all ciphers, in µs.

Memory Usage

Another important notion when talking about resource-constrained devices is the memory available. Lightweight cryptography tends to focus both on optimizing the power consumption while encrypting and decrypting and developing solutions that take up the minimum amount of resources available without

compromising the security of the cipher. Since Arduino Uno is the device available for this study with the fewer computational specifications available we will also analyze how each cipher optimizes its memory usage to fit into the available space. In Figure 6.6 we can see the SRAM and Flash Memory used per cipher on each implementation. Considering that the maximum available Flash Memory is 32000 Bytes and the maximum SRAM is 2000 Bytes, we can understand the margin available after compiling each program. Analyzing the former, XTEA and ASCON occupy the biggest percentage of available memory, at 41.2% and 40.5% respectively. Then AES uses 29.4% and, distinctively below any of these three ciphers, TEA with only 16.3%. Given its extremely lightweight nature and lower complexity, it is understandable how it manages to occupy such a small piece of memory, especially in comparison with the other 3 approaches.

In contrast, SRAM's usage is the highest in AES, at 76.4%, followed by XTEA and ASCON at 53% and 52.6%, respectively. The only constant is TEA which, once again, uses by far the lowest amount of memory, at just 15.2%. These values are extremely important to understand the margin available, since many of these constrained devices require either further functionalities or bigger inputs, both of which will increase the memory load.

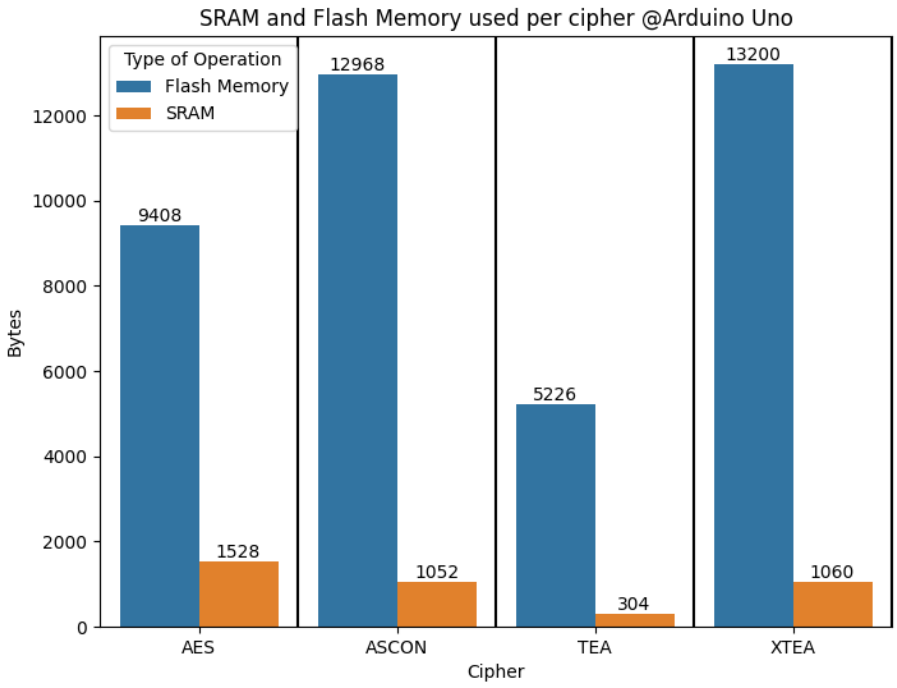


Figure 6.6: SRAM and Flash Memory used by each cipher on an Arduino Uno.

Different Modes of Operation

This sub-section is comprised of both time and power analysis of similar ciphers using different modes to assess how their construction affects their overall performance.

As already stated, all of the tested ciphers are using the ECB mode of operation, to maximize their efficiency. However, given that some systems might require increased resilience or authentication, variants of AES using GCM construction and XTEA using CFB construction were used and tested to assess their performance differences. Although the mode of operation is different, the implementation comes from the same source, in order to minimize any anomalies while testing that could be attributed to different implementations. In Figure 6.7 we have the time required, in µs, to perform one encryption and decryption of the different modes. Unlike ECB, both GCM and CFB are much closer to each other, drastically reducing the difference when performing these two operations. When it comes to the time needed, when encrypting, the difference is quite significant for AES which, on average, is 185.3% faster using ECB, although the difference reduces to 47.8% whilst decrypting. This sharp increased aligns with the notion that both the increased complexity and added authentication substantially affect the required time to perform these operations. As for CFB, the values for both encryption and decryption are identical, although a bit slower than ECB which, on average, is faster by 4% with encryption and 8% with decryption, a greatly reduced margin especially comparing with GCM.

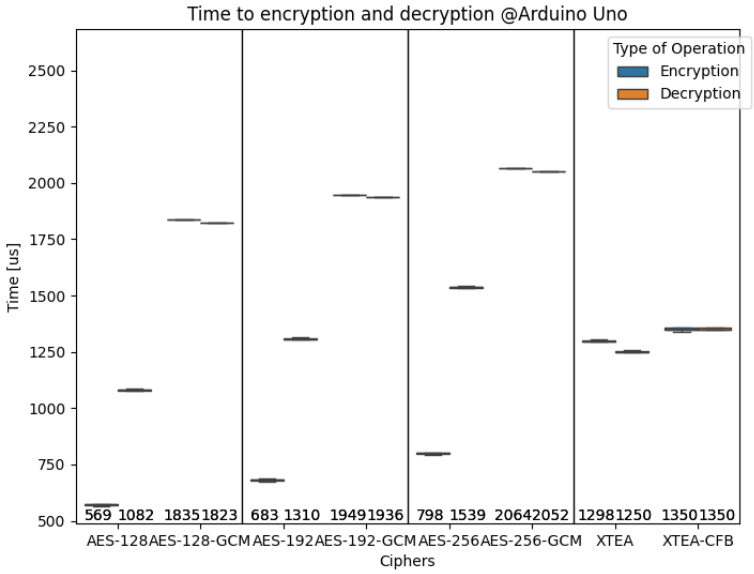


Figure 6.7: Time per encryption and decryption for different modes of operation.

When it comes to power consumption, Figure 6.8 shows the values, measured in Watts, that compare the different modes of operation. Analysing AES and its variants, GCM requires some additional power per operation due to its increased complexity however, the difference is not as significant as the time measurements showcased in the previous Figure 6.7. CFB displays the exact opposite behavior, being faster than ECB whilst being used with XTEA. Although relevant to the study, we already highlighted the shortcomings of Arduino when it comes to accurately assessing data for power, given its resource-constrained nature and very low susceptibility to variations in energy consumption, so this study will also be conducted in an ESP32, which will be showcased on the next sub-chapter.

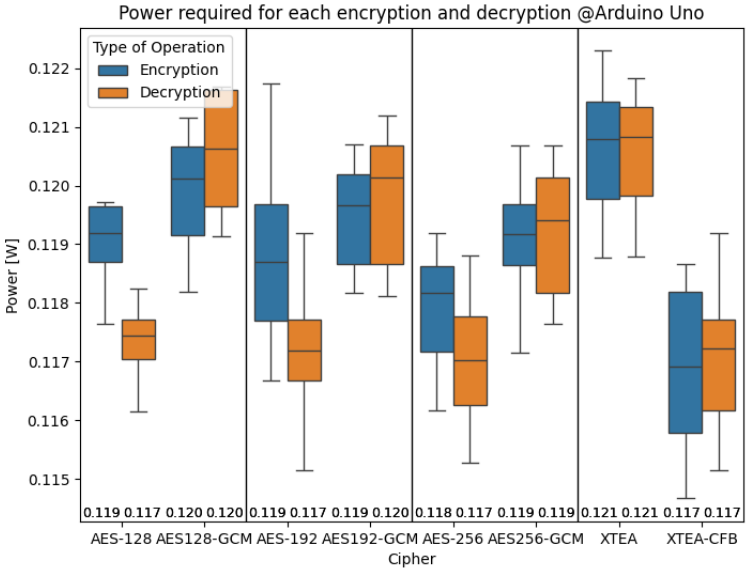


Figure 6.8: Energy consumption for the different modes of operation.

6.3.2 ESP Devices

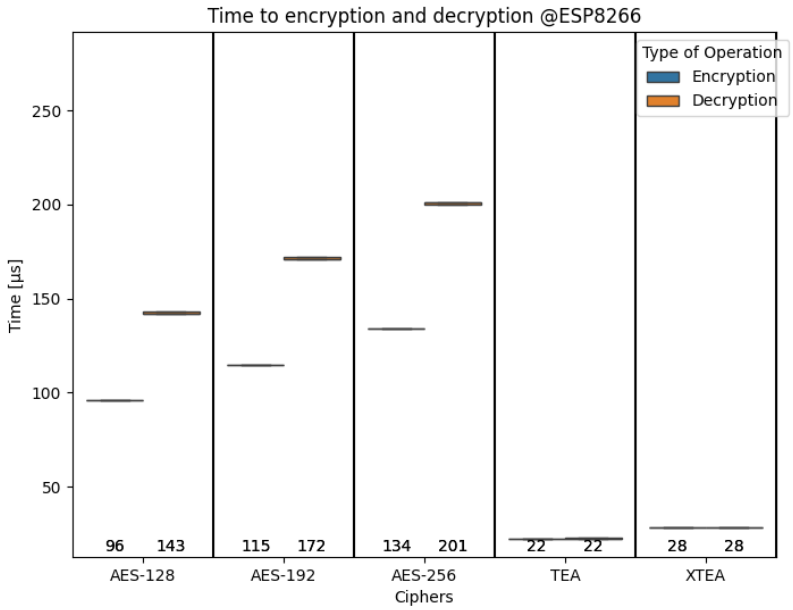
This section will provide all of the data gathered about both ESP devices, as well as a critical analysis of the performance of each cipher when compared to one another. One important note that should be mentioned is that given a multitude of errors and incompatibilities, no version of ASCON was successfully validated for both ESP32 and ESP8266, so this section will not include any data into either of them.

Speed and Power Outputs

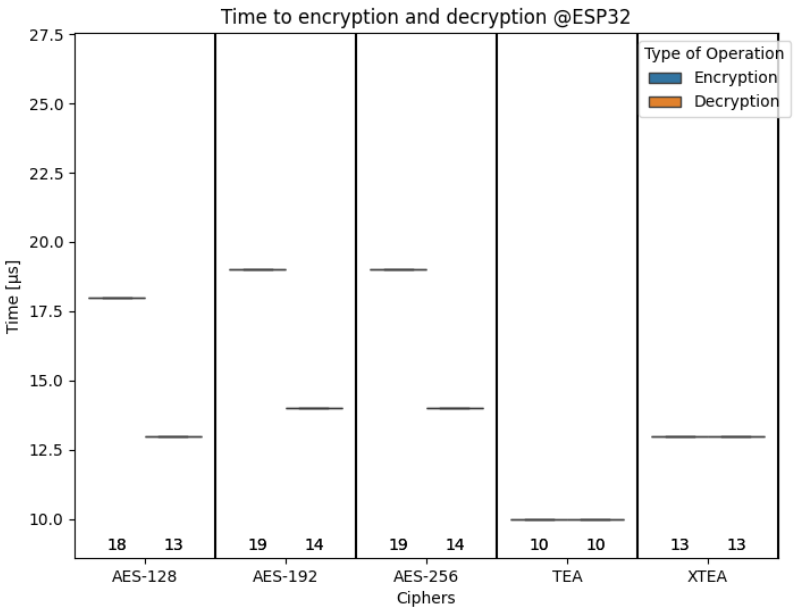
Given their different technical specifications, the two ESP devices subject to analysis in this thesis present distinct speeds while performing their operations and draw different power amounts. Looking at Figure 6.9, this contrast is already noticeable, with both times being drastically shorter on the ESP32, given its higher clock speed. Comparing each cipher, on ESP8266 the same trend as Arduino Uno can be seen, progressive increase of time as the key size of AES increases and a distinct value between encryption and decryption, with the latter taking considerably more time, 57us and 49.5% more to be specific for the three variants, on average. TEA and XTEA take substantially less time to perform both operations (more than 5 times faster than the three AES on average) but still maintain their order the same way as in Arduino.

However, ESP32 appears to contradict these notions, with the times for both encryption on AES being relatively the same, instead of increasing, and, more crucially, with the decryption being faster than the encryption on all three versions by 36.5%. Furthermore, TEA and XTEA are only marginally faster than AES for

decrypting, although while encrypting each byte they were, on average, 44.5% and 27.8% faster, respectively. Another key point from this discussion is the variation, as noticeable, all values measured are extremely close to the average, with minimal variation on both devices.



(a) ESP8266



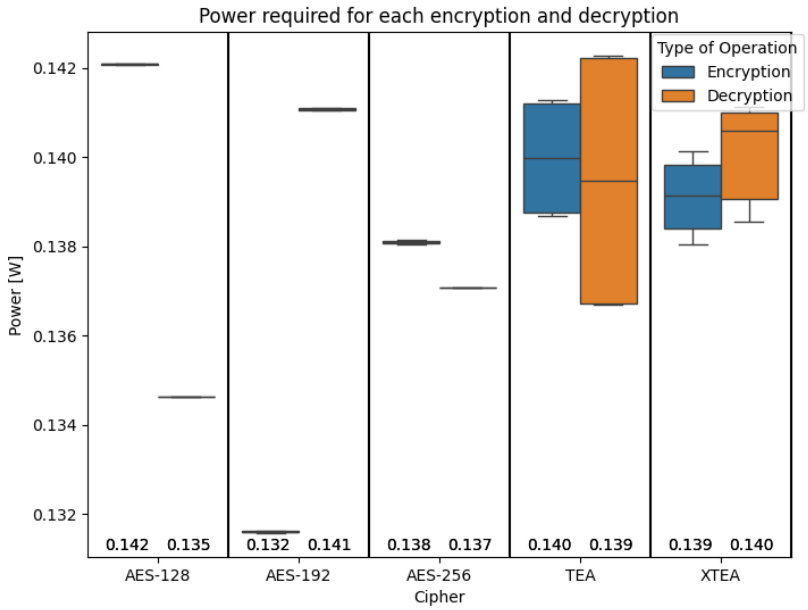
(b) ESP32

Figure 6.9: Time per encryption and decryption on both ESP devices.

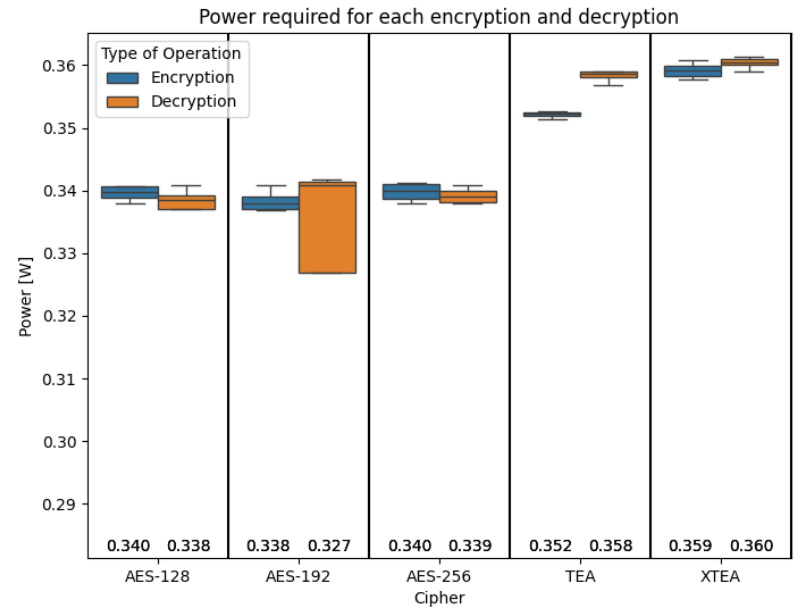
Moving on to the power consumed by each cipher on the ESP family, Figure 6.10 depicts the energy used by each cipher to perform one encryption and decryption, on average. Starting with ESP8266, one noticeable aspect would be the variation displayed by both TEA and XTEA which contrasts with the little to no variation for the variants of AES. Furthermore, on the latter, encrypting is faster than decrypting for the 128 and 256 key size variants but the opposite occurs with the 192 bits version. For enhanced stability between these two operations, both TEA and XTEA work best, with close results on average despite their increased variation of results.

As for ESP32, both encryption and decryption are much closer to each other on all ciphers. However, when it comes to the variation of results on each column, AES now presents a slightly higher number when compared to TEA and XTEA, the exact opposite of ESP8266. Another important notion would be the higher energy consumption for the latter two ciphers, requiring 4.8% and 7.2% more for both encryption and decryption in contrast with AES and its three key-size variants.

Figure 6.11 displays the power latency of each cipher for both devices. Again, for both devices, AES-192's decryption seems to be an outlier, not following the gradual ascendant trend from the lower key size to the higher. Other than that, the results are quite similar across the two ESP devices, with decryption taking up more energy per bit than encryption for the AES variants, while it is the opposite for both TEA and XTEA. To maximize energy saving per bit of data, AES-128 would be the best choice to perform encryption, however, given its exceedingly large energy cost for decryption, a more nuanced and close approach could work best, by for example, choosing TEA/XTEA or taking advantage of the reduction present within AES-192's decryption.

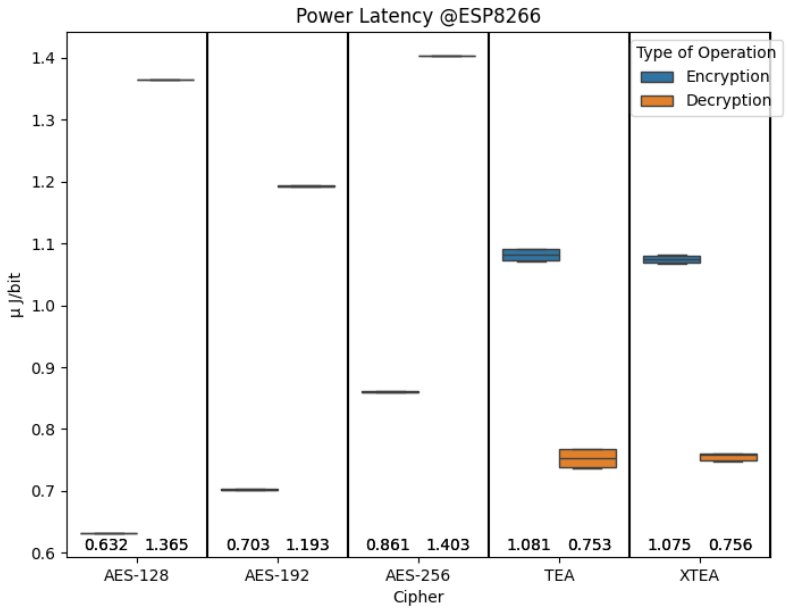


(a) ESP8266

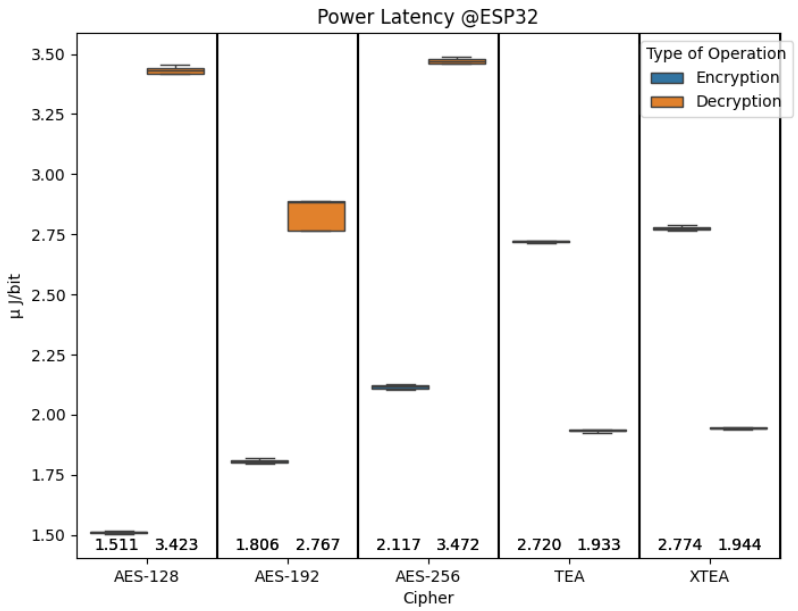


(b) ESP32

Figure 6.10: Power consumption in Watts on both ESP devices.



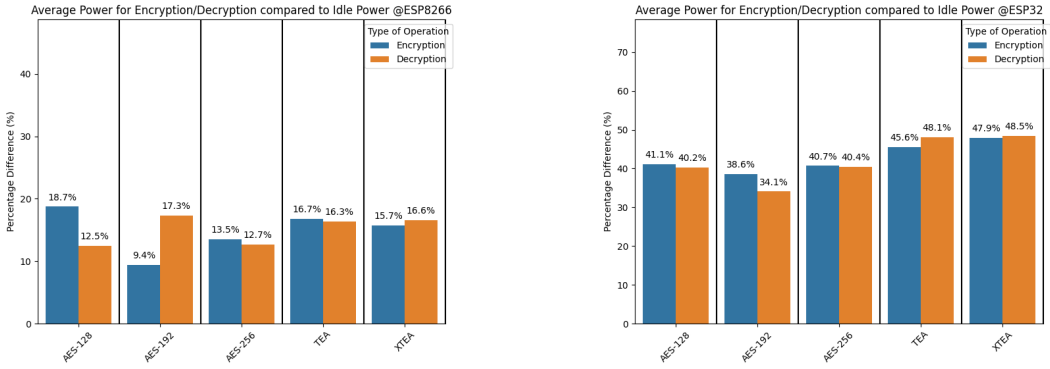
(a) ESP8266



(b) ESP32

Figure 6.11: Power latency in µJ per bit on both ESP devices.

As for Figure 6.12, more details are provided on the difference in power consumption when performing both encryption and decryption versus an idled device. Starting with ESP8266, it is once again noticeable how close TEA and XTEA are while performing both of these operations, around 16%. This, of course, contrasts with the AES variants which have a more steep difference between encryption and decryption. However, when analyzing ESP32's data, there is a more significant increase in power consumption, which, on average, is around 40% for AES and 47.5% for TEA and XTEA, far greater than the observed values in ESP8266. We believe that these data begin to provide support for the theory that, with increased resources, the usage of energy gradually increases as well when compared to an idled device.



(a) ESP8266 (b) ESP32

Figure 6.12: Power consumption difference compared to idle power on both ESP devices.

Different Modes of Operation

The plan of analysis here is similar to Arduino Uno, with ECB and GCM being used for AES and ECB and CFB as the alternative for XTEA.

Starting with the time per operation, Figure 6.13 highlights, once again, the discrepancy between the two modes of operation, with ECB, on average, 57.5% faster than GCM whilst encrypting and 132.3% when it comes to decrypting data. Another interesting conclusion is that the variation for both operations is not as significant as it was with Arduino Uno, with decryption being considerably slower than the encryption for ECB and the opposite when it comes to GCM. As for XTEA, again, the difference is small between both modes and they take, on average, the same time to perform both the encryption and decryption.

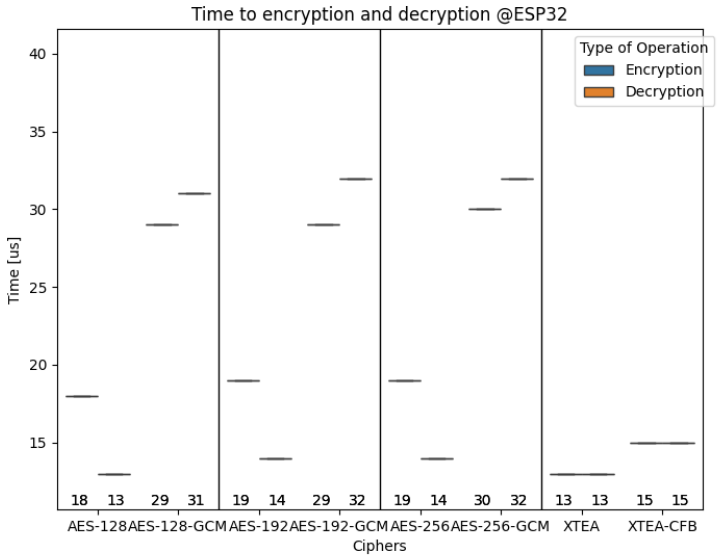


Figure 6.13: Time per encryption and decryption for different modes of operation.

As for the power consumption, in Figure 6.14 we have a graphical representation of these values for the same ciphers. Starting with AES, this data corroborates the findings of Arduino Uno; beyond being faster, ECB also draws less power on all different key sizes compared to GCM, more specifically, 5.3% and 7% less with encryption and decryption, respectively. There are also some noticeable variations in these values, although excluding AES-128 using ECB, it isn't excessive. The biggest question mark would be how XTEA's power consumption would fluctuate with the different modes used, as Arduino Uno wasn't able to provide a clear view of this. The results are more even this time, with decryption taking up the same energy and encryption being slightly less consuming, at 1.7%.

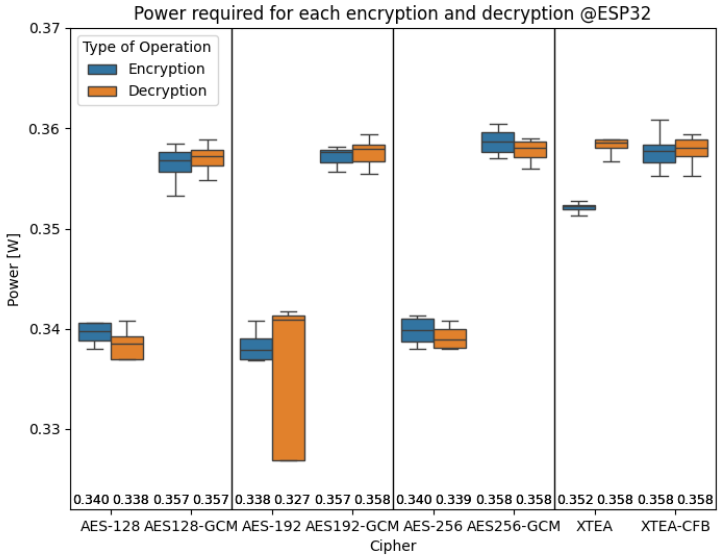


Figure 6.14: Energy consumption for the different modes of operation.

6.3.3 Raspberry Pi Devices

This section presents all the data gathered on the three Raspberry Pi devices, as well as a critical analysis of the performance of each cipher in comparison to the others.

Speed and Power Outputs

To start the analysis, we will begin by studying the time needed for encryption and decryption on the Raspberry Pi 3B+. As present in Figure 6.15, we can point out, again, the correlation between an increased time and key size for AES. Furthermore, ASCON, TEA, and XTEA are significantly faster than any of the AES variants, by a margin, on average, of 51.9%, 70%, and 61.9%, respectively, for encryption and 59.9%, 70.6%, and 62.5%, respectively, for decryption. There is some variation in the obtained values for all ciphers, excluding TEA and XTEA.

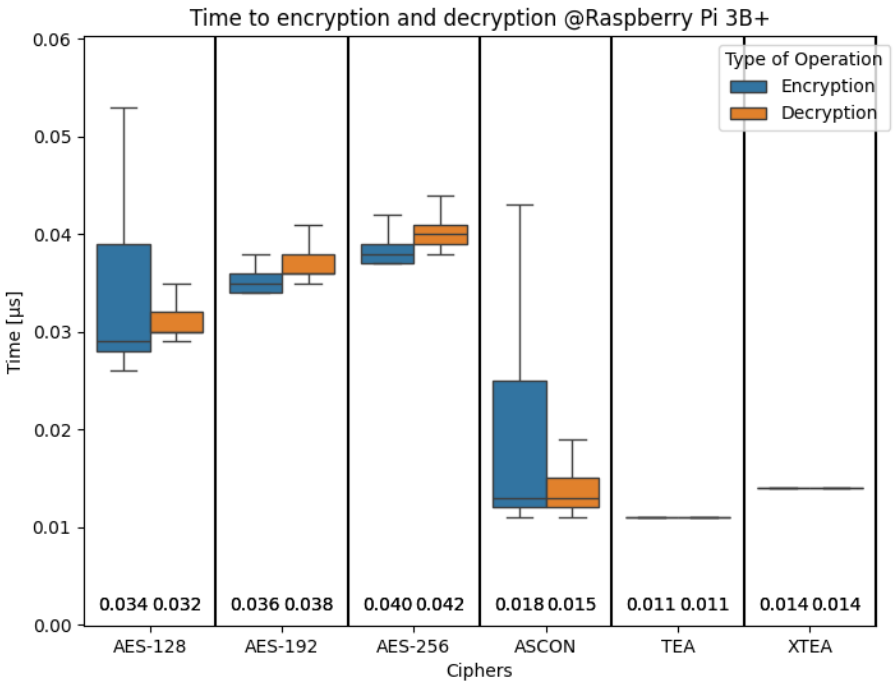


Figure 6.15: Time in µs for one singular encryption and decryption, for different ciphers on a Raspberry Pi 3B+.

Figure 6.16 showcases the time necessary to encrypt and decrypt in the Raspberry Pi Zero, a device with similar technical capabilities to the ones of the Pi 3B+. The similarities are vast, with, once again, the proportional increase of both key size and time per operation on AES, and ASCON, TEA, and XTEA being very closely matched and much faster than any AES variant. There is also a noticeable variation for all ciphers, minus TEA and XTEA. Analyzing how fast ASCON, TEA, and XTEA are, on average, when compared to AES yields the following values:

56.7%, 68.3%, and 62.5%, respectively, whilst encryption, and 63.9%, 69.5%, and 63.9%, respectively, upon decryption.

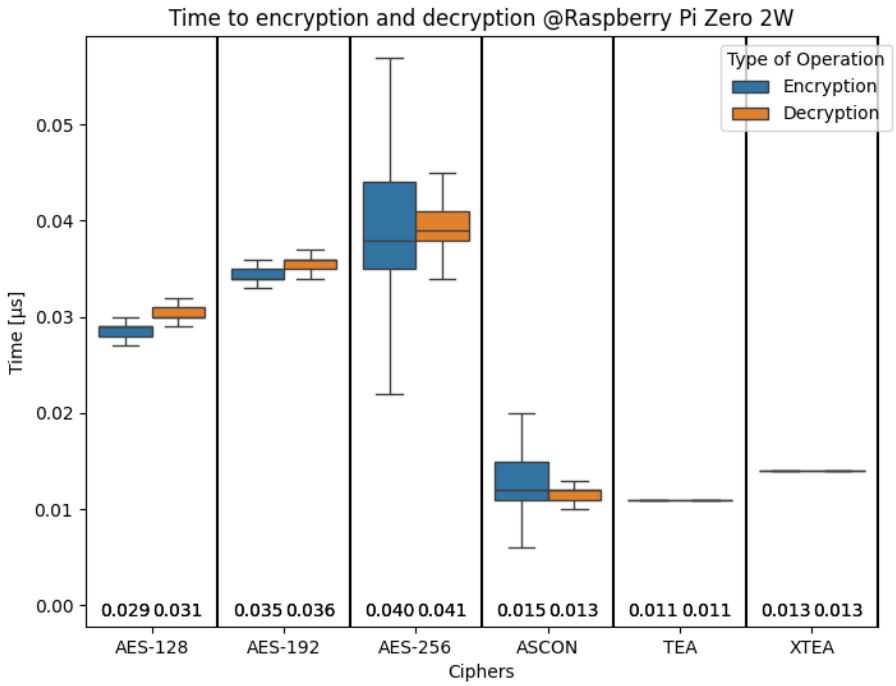
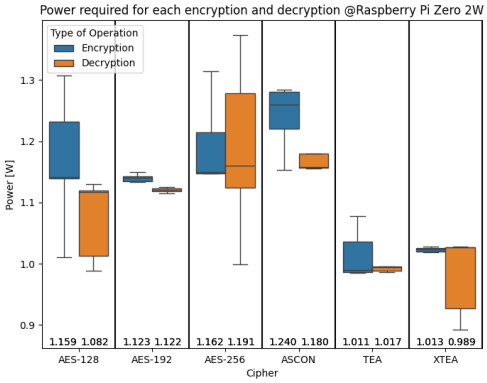


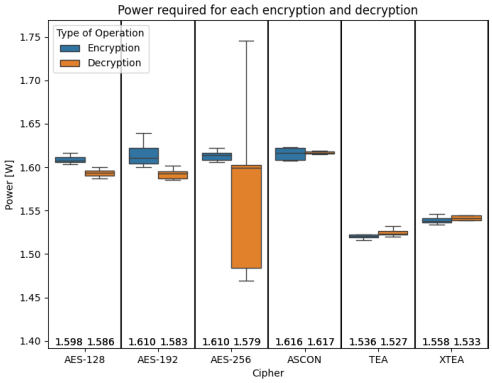
Figure 6.16: Time in µs for one singular encryption and decryption, for different ciphers on a Raspberry Pi Zero.

Moving on to the power analysis, Figure 6.17 showcases the power consumption for the three Raspberry Pi devices subject to test in this thesis. The first thing to point out would be the different categories of values displayed by the Pi Pico and the other two devices, given that the former possesses less computational capabilities than the latter two, thus contributing to greater power saving. As for each cipher, it is harder to pinpoint patterns that are consistent across the three devices, TEA and XTEA, consume less energy for the two most demanding devices, 3B+ and Zero, by some margin, however, when used in the Pico, they, together with ASCON, consume far more power than AES. Another noticeable difference between the group of 3B+ and Zero, and Pico, would be that decryption takes up less energy for AES, however, this notion is once again reversed in the latter device, again, by a considerable margin.

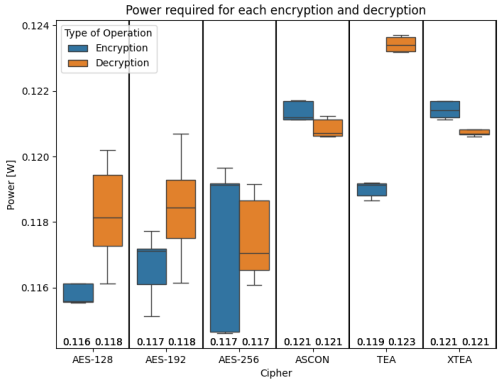
Figure 6.18 depicts the power latency, measured in µJ per bit, for both the Pi Zero and the Pi 3B+. A stark contrast is noticeable between encryption and decryption for the three AES variants, with decryption taking up almost double the energy per bit of decryption when compared to the encryption, on both devices. Furthermore, ASCON, TEA, and XTEA are all closely matched, with the former being slightly better at optimizing its energy use than the latter two. Another noticeable trend with these three ciphers is how close both encryption and decryption are to one another.



(a) Raspberry Pi Zero

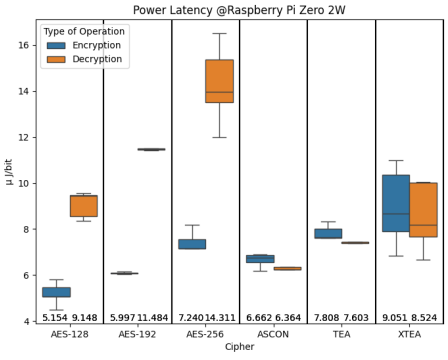


(b) Raspberry Pi 3B+

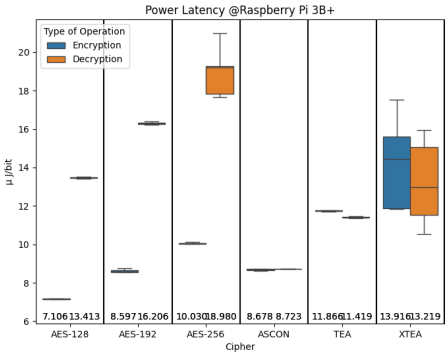


(c) Raspberry Pi Pico

Figure 6.17: Power consumption in Watts for the three Raspberry Pi.



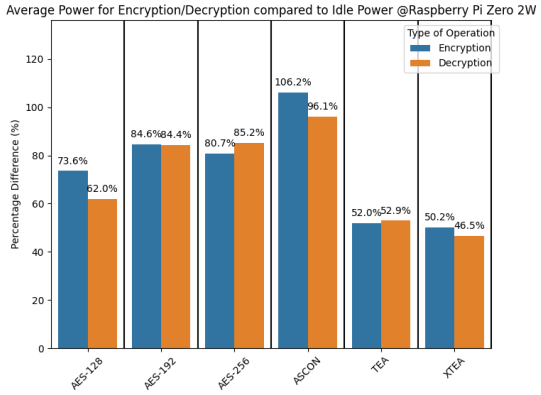
(a) Raspberry Pi Zero



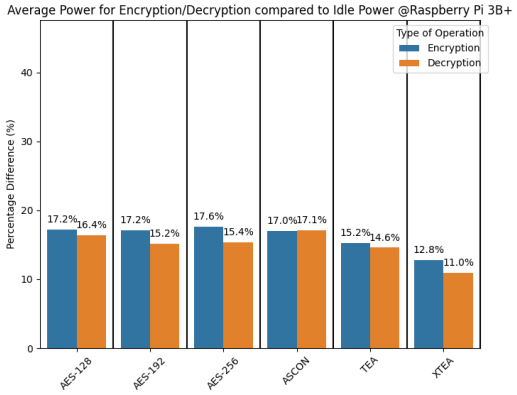
(b) Raspberry Pi 3B+

Figure 6.18: Power latency in µJ per bit for Pi Zero and Pi 3B+.

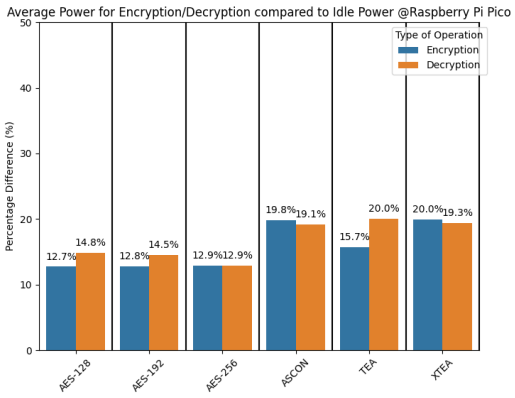
The last idle comparison, as showcased in Figure 6.19, displays the contrast between the three Pi devices. A noticeable anomaly would be the extremely high difference between idle power and encrypting/decrypting for the Pi Zero, which on the low end is 50% above and on the high-end 106%. Not one Raspberry device, or another discussed in previous chapters comes close to these values, with those ranging from 10% to 20% above idle power.



(a) Raspberry Pi Zero



(b) Raspberry Pi 3B+



(c) Raspberry Pi Pico

Figure 6.19: Power consumption difference compared to idle power.

Different Input sizes

Once again, tests were also conducted on Raspberry Pi Pico with different input sizes to complement the work on the Arduino Uno. Table 6.2 showcases those values. Unlike Uno, both times for TEA are the same, no longer exhibiting a noticeable discrepancy. Furthermore, ASCON and XTEA still diverge somewhat, although it is noteworthy that the variations present in this table are more drastic than those depicted in Table 6.1.

Metrics	AES-128	AES-192	AES-256	ASCON	TEA	XTEA
Time (µs)	56	66	76	18.18	11	157
Time Adjusted (µs)	56	66	76	36.36	22	39.25
Time for 16 Bytes (µs)	56	66	76	40	22	35
Buffer (Bytes)	16	16	16	8	8	64

Table 6.2: Difference between time and time adjusted on all ciphers for Raspberry Pi Pico, in µs.

6.4 Device comparisons

In this section, the goal is to compile all the data that we already have and establish a fair assessment of each device on both their time and power used per encryption and decryption, on average, across all ciphers.

6.4.1 Differences in Speed

Figure 6.20 showcases each device's performance, in µs, while performing both operations. Given its few computational resources, it is no surprise that the Arduino Uno stands distant from all the others, with 837 µs and 1138 µs of encryption and decryption time, respectively. Pico and ESP8266 are somewhat close to each other, with the latter being slower than the former, although both are still around 10 to 15 times faster than the Arduino Uno. ESP32 follows suit, with 15 µs and 12 µs required to complete each operation, and, lastly, Zero and 3B+ are close to each other, at 0.024 µs and 0.026 µs respectively. The line that correlates their speed capabilities and frequency speed is proportional on all devices except Zero and 3B+ which, surprisingly, does not follow this order, with Zero being faster despite having a clock speed 400MHz slower than the former. This might be due to some optimization issues on the 3B+, which might mismanage how efficiently the processor is used and thus hinder its performance.

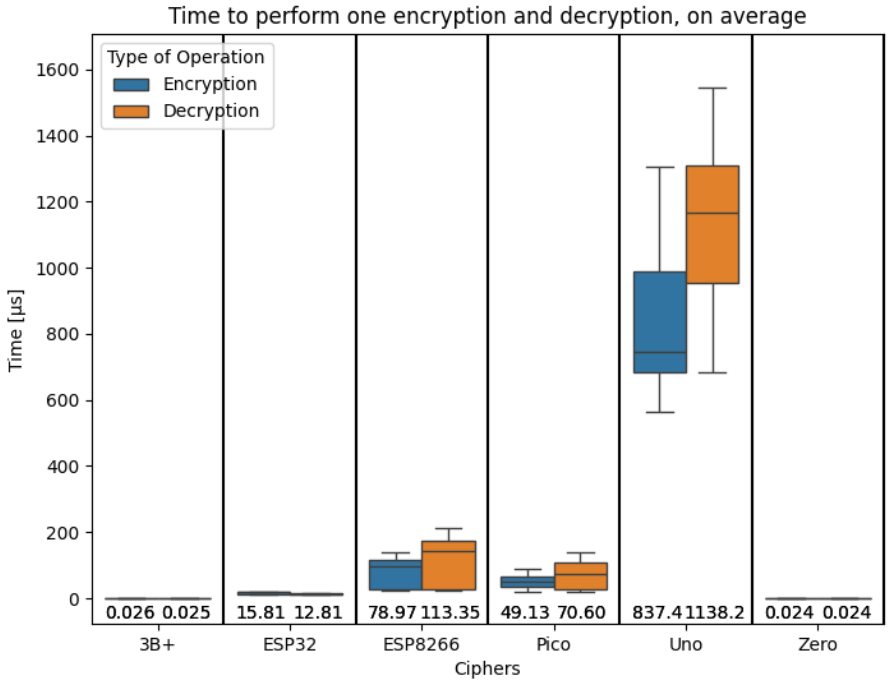
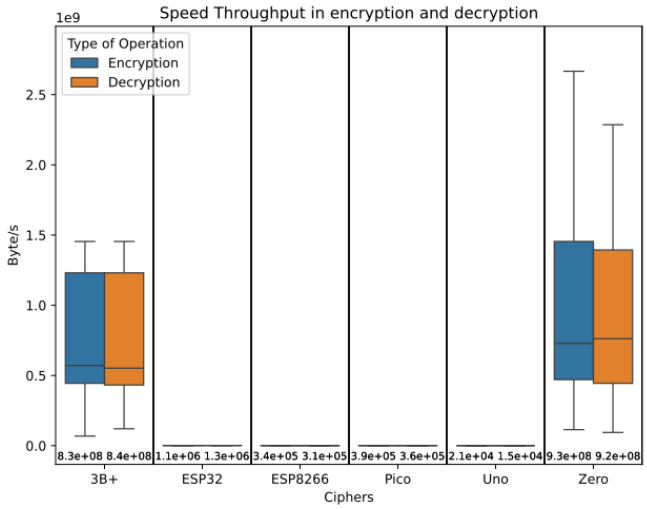
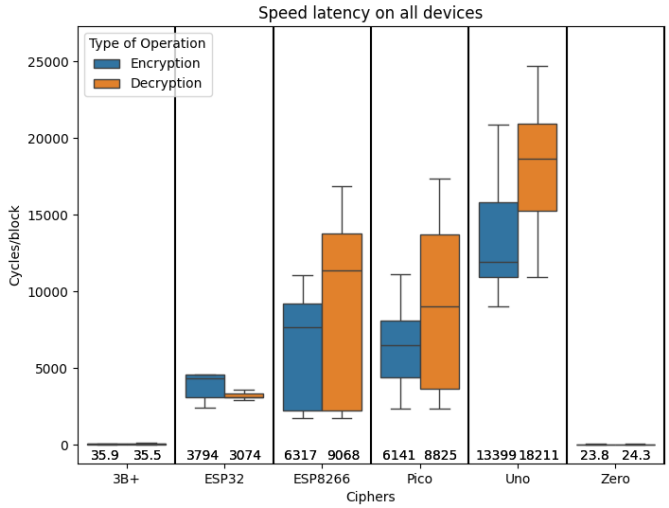


Figure 6.20: Time in µs for one singular encryption and decryption for all devices.

Continuing our analysis, in Figure 6.21 we have both the speed throughput and latency for each device. As expected, the more resource-capable mechanisms can process significantly more bytes per time unit than other more limited ones, such as the Arduino Uno or ESP8266. Focusing on the speed latency, which translates to cycles per block, this visualization takes into account each clock speed to translate the amount of cycles needed to encrypt each block of data which, as we know, varies from cipher to cipher. As expected, both 3B+ and Zero, given their speed, take far fewer cycles to encrypt each block, which heavily contrasts with the other devices, as Arduino Uno takes, approximately, 400 to 500 times more cycles to encrypt the same block of data as these two devices.



(a) Speed Throughput



(b) Speed Latency

Figure 6.21: Speed throughput and latency for all devices.

6.4.2 Differences in Power Consumption

Moving on to the power analysis, we hope to see a similar line of causation between the resources available and the power drawn. Looking at Figure 6.22, all results are consistent with their computational capabilities apart from two devices, Pico had less power consumed per encryption than Arduino. Another important notion that was mentioned throughout the analysis of the ciphers is that the devices with fewer resources also present fewer variations in their measured power values, which is noticeable if we look at ESP8266, Pico, and Arduino, the three low-end devices with almost no variation, and compare them with ESP32 and the two remaining Raspberry Pi hardware, which showcase this variation as it is noticeable in their value variation.

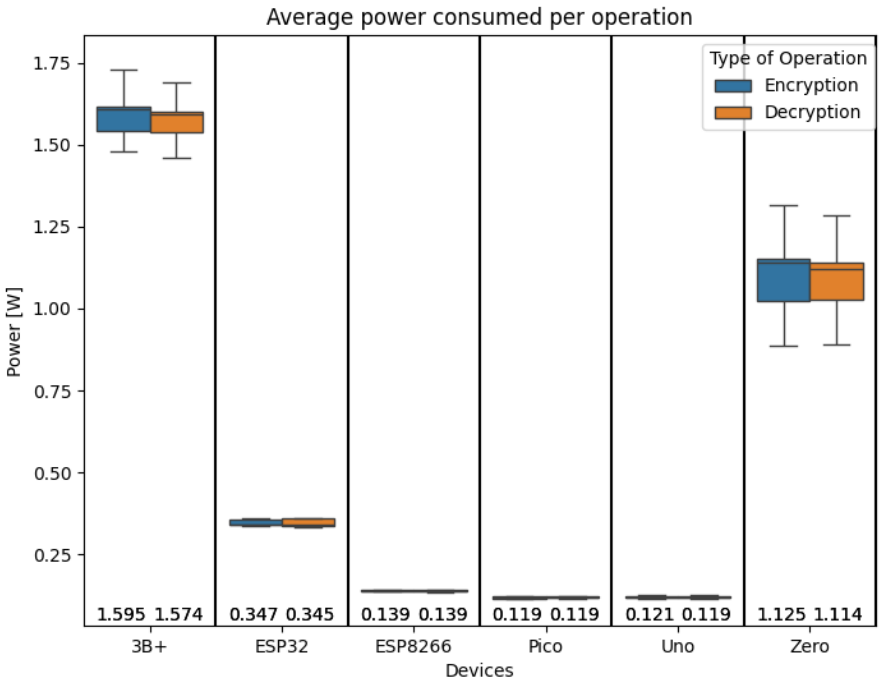


Figure 6.22: Average power consumption measured for all devices.

6.5 Scenarios for their application

One of the cornerstones of this thesis is not only to increase the available knowledge on lightweight encryption performance using validated metrics and widely used IoT devices but also to provide applicable situations in which the gathered data can be put into practice to optimize current solutions on seaport's use cases. With that in mind, it is important to understand three key requirements before drawing out any potential application scenarios.

- **Speed:** The necessary speed rate on which data needs to be encrypted. It can vary from not relevant when dealing with low transmission rate scenar-

ios or with tolerance to delays to extremely important, especially to support automation or monitor real-time data with little to no resistance to delays.

- **Resources:** The amount of resources available to each device for its operation, which include computational capabilities and energy capacity. It can range from very low when dealing with an Arduino Uno or battery-powered devices to very high if the device has vast resources, as is the case with the Raspberry Pi Zero or 3B+.
- **Accessibility:** This criteria establishes whether the IoT technology to be used is easily accessible or not, based on their positioning and geo-location. The lower the accessibility, the more costly and time-consuming it is to replace its battery or a device altogether.

The following list will contain some scenarios, based on the knowledge present in the state of the art concerning seaports and lightweight technology, on which to apply the data gathered in this chapter to establish some guidelines on which cipher and device to use given a set of requirements.

A1: Device to capture sea conditions

Description: Several battery-powered devices would be placed on buoys at different points of the sea, somewhat near the seaport to record important data such as sea current, wave height, and sea temperature to optimize the traffic network and provide accurate information to the relevant parties.

Requirements:

- **Power Usage:** Very Low.
- **Speed:** Not critical.
- **Accessibility:** Not easily accessible.
- **Security Concerns:** Low.

Device and Cipher Chosen: Given the necessity for power saving since the devices run on battery power and are not in an easily accessible place, an Arduino Uno would be the best choice for its extremely low energy consumption when compared to the other devices. As for the cipher used, going back to the appropriate figures showcases that AES consumes less energy in both Watts and µJ per bit of encryption, and, given that a heavily secured system is not essential in this scenario, the 128-key size variant would suffice.

A2: Monitor whether a parking spot is empty or occupied

Description: Several battery-powered devices would be positioned next to parking spots on the seaport and, using sensor technology would detect whether it is available or not, as well as, if needed, provide other data like image information.

Requirements:

- **Power Usage:** Low.
- **Speed:** Not critical.
- **Accessibility:** Somewhat accessible.
- **Security Concerns:** Medium.

Device and Cipher Chosen: Given the previously mentioned requirements, probably the most suited device would be medium-end hardware like ESP8266 that whilst using low power, can reliably transmit the needed data, although Arduino Uno or a Raspberry Pi Pico could also be other decent choices if the only goal were to transmit simple data on whether the spot is empty or not. As for the encryption cipher, given the need for some security given the imaging capabilities, an efficient yet reliable cipher is needed, which is why XTEA would be a great choice, as it works very efficiently when compared to AES and yet, whilst only marginally slower than TEA, possesses a greater notion of security, essential when dealing with this type of data.

A3: Monitor structural integrity of cranes

Description: Some battery-powered devices would be placed on critical points of a crane to study their potential fatigue and identify possible cracks that could jeopardize their security and workability.

Requirements:

- **Power Usage:** Very Low.
- **Speed:** Not critical.
- **Accessibility:** Not easily accessible.
- **Security Concerns:** Medium.

Device and Cipher Chosen: The need to use the lowest amount of power possible to conserve as much battery life as possible makes the choice of Arduino Uno the best possible one due to its extremely low power consumption. To complement this, AES-128 should also be used alongside the hardware as it fits within the security requirements whilst delivering a low-power approach, especially when compared to variants with higher key sizes.

A4: Capture data to support autonomous movement of cars in the seaport

Description: Several devices, powered by the autonomous car they are in, track the precise location of their vehicle and scan for any nearby obstacles that can compromise their route. This data is quickly sent to a control unit to process it.

Requirements:

- **Power Usage:** Not relevant.
- **Speed:** Extremely critical.
- **Accessibility:** Easily accessible.
- **Security Concerns:** High.

Device and Cipher Chosen: In this scenario, both speed and security are the most critical requirements so, in accordance, the recommendation of the device to be chosen would be a Raspberry Pi Zero or 3B+ given their efficiency when encrypting and decrypting data. Nonetheless, an ESP device could be chosen if the Raspberry Pi's computational capabilities and graphical interface exceed the scenario necessities by a considerable margin. As for the chosen cipher, given the results for time and speed gathered, TEA or XTEA could be seen as the most logical choices, however, the former must not be inserted in this process due to its inherent security concerns and, since the data transmitted can compromise the safety of the vehicles and even nearby people, only XTEA with its more security-driven nature should be considered, as it provides the necessary protection and underlying efficiency.

A5: Receive data and create security backup

Description: A device receives data from multiple activities within the seaport and creates a secure backup of them on an outside database.

Requirements:

- **Power Usage:** Not relevant.
- **Speed:** Not critical.
- **Accessibility:** Easily accessible.
- **Security Concerns:** High.

Device and Cipher Chosen: Given the complexity of the task, the most obvious choice would be a high-end device, namely Raspberry Pi Zero or 3B+. Since there is no immediate need for speed and it is not critical to keep power usage to a minimum, although still desirable to optimize the seaport's power consumption, there is no ideal cipher to choose from, apart from TEA given its security

compromises. The advised solution would be more concerned with the mode of operation of the cipher chosen, which means that if AES is selected, it would need to run on GCM mode instead of ECB given the security concerns of the latter. If, on the other hand, XTEA was to be selected, CFB would be the desired mode of operation. Furthermore, for the cipher chosen, one should also take into account the compatibility with other devices, the main job to perform here would be decryption if we're taking into account that any data received is already encrypted and needs to be accessed, so that should also be a priority in this case.

A6: High flow of data to coordinate seaport management

Description: A complex singular device that coordinates multiple systems in the seaport based on a variety of data received from several sources.

Requirements:

- **Power Usage:** Not relevant.
- **Speed:** Extremely critical.
- **Accessibility:** Easily accessible.
- **Security Concerns:** High.

Device and Cipher Chosen: Once again, since this scenario demands a highly capable and resource-driven device, it would either be a Raspberry Pi Zero or 3B+ since these are the only ones that can achieve the required technical capabilities to perform this task. As for the cipher chosen, it should be similar to the previous scenario, with a similar focus on compatibility with the encryption standards used by other devices. Apart from TEA due to its inherent security concerns, all of the other ciphers could adapt to the given scenario.

The following Table 6.3 summarizes the presented scenarios to easily understand the best choices for each situation.

Scenarios	IoT Devices						Ciphers					
	Uno	ESP8266	ESP32	Pico	Zero	3B+	AES-128	AES-192	AES-256	ASCON	TEA	XTEA
A1	✓	×	×	×	×	×	✓	×	×	×	×	×
A2	✓	✓	×	✓	×	×	×	×	×	×	×	✓
A3	✓	×	×	×	×	×	✓	×	×	×	×	×
A4	×	×	✓	×	✓	✓	×	×	×	×	✓	✓
A5	×	×	×	×	✓	✓	✓	✓	✓	✓	×	✓
A6	×	×	×	×	✓	✓	✓	✓	✓	✓	×	✓

Table 6.3: Summary of devices and ciphers per scenario.

Chapter 7

Assessment of Transmission Standards

The seventh chapter contains all the data gathered on different transmission standards used in a IoT and seaport context. Section 7.1 demonstrates the detailed work plan for this chapter, as well as all the tests to be done and their parameters. In Section 7.2, several comparisons are established between each standard, based on their power consumption and transmission rate. Section 7.3 complements the previous section by adding range tests performed on some standards to understand how far they can transmit data and at which strength, which is crucial to understand whether they are appropriate for each proposed scenario. Lastly, in Section 7.4 the scenarios drawn out in the previous chapter will be applied here with the conclusions of each transmission standard, to understand how the encrypted data can be transmitted more efficiently and safely.

7.1 Testing Campaign

Given that we need boards that support wireless data transmission, Arduino Uno and Raspberry Pi Pico will be excluded from this assessment given their lack of hardware compatible with the transmission standards at hand. This leave us with the two ESP devices and Raspberry Pi Zero and 3B+. Of these four, all but ESP8266 support Bluetooth, and the latter two do not have ESP-Now, since this standard is only used by ESP family devices. The data transmitted by each standard as part of the testing routine were 16 bytes of encrypted information by TEA, although standards like ESP-Now carry this information in a larger block of 250 Bytes due to its packet structure.

Starting with Bluetooth, these tests will be carried out for ESP32, Raspberry Pi Zero, and 3B+, since these are the only devices with a built-in module for Bluetooth. Their communication will be ensured with a smartphone, using a Serial Bluetooth Terminal application capable of establishing communication with the IoT devices and exchanging information with them. Figure 7.1 depicts this relation, with both the Classic and low energy BLE versions being subject to testing

for the ESP32, while Bluetooth Classic is being used for the two Raspberry Pi.

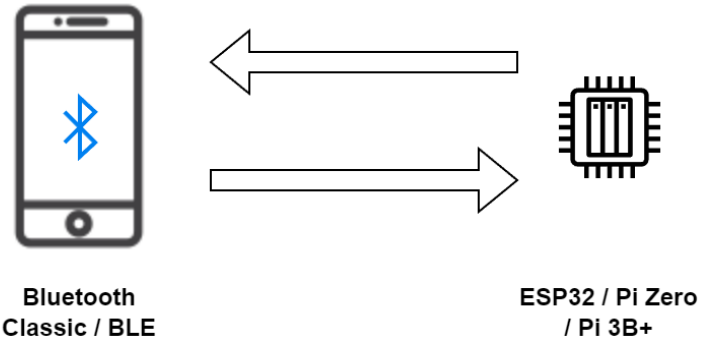


Figure 7.1: Routine of tests for Bluetooth Classic and BLE.

Next, we have the Wi-Fi tests, which will be divided into two components as noticeable in Figure 7.2, one between the ESP devices and a smartphone, with the latter acting as the Access Point, and a secondary test on which there is a direct connection via Wi-Fi between the two devices, to test their power consumption while being used as an Access Point or simply a client connecting to it. It is noteworthy to mention that the data collected from the first component will be useful to assess the range parameters and how its Received Signal Strength Indicator (RSSI) [1] and subsequent signal decay and maximum range can be defined for this standard.

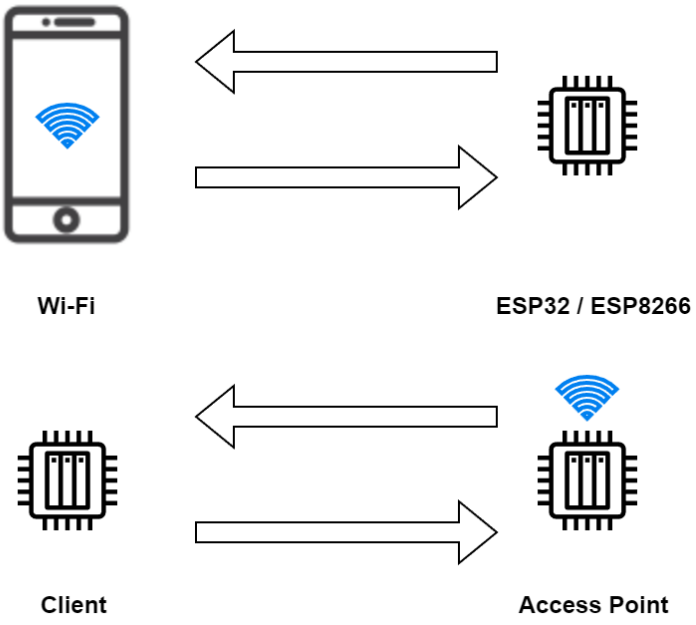


Figure 7.2: Routine of tests for Wi-Fi.

Lastly, there will also be performed tests on the ESP-Now transmission standard, as depicted in Figure 7.3. Since this standard only works with ESP hardware,

[1]The RSSI is a measurement of the intensity of signal present between two devices, being useful to determine the degradation of signal with increased distance [Metageek]. These values are measured in dBm and usually range between -30 and -100, with the latter representing an optimal RSSI and the latter an almost loss of signal

only ESP32 and ESP8266 will be subject to this test routine. The goal is to assess the power consumption whilst sending and receiving data on each device and get a quantifiable measure of the available range and subsequent signal quality. These tests will be carried out exclusively with the support of Arduino IDE and the code provided in the documentation for ESPNOW by its manufacturer and creator Espressif Systems.

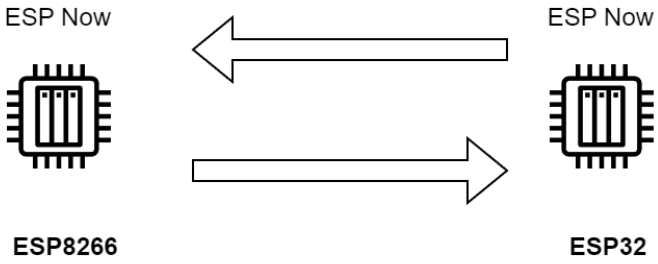


Figure 7.3: Routine of tests for ESP-Now.

7.2 Transmission Rate and Power Analysis

To start our analysis, we'll use ESP32. In Figure 7.4 we have several a power consumption comparison between Bluetooth Classic, BLE, and ESP-Now. Beyond this, each standard was also tested in two modes: "Slow", which depicts a slower transmission rate (one packet per second), and "Fast" which correlates with the fastest transmission rate available for each standard.

To start, the increase in power consumption as the transmission rate is higher is noticeable across all transmission standards, albeit how much varies greatly from one to another. Bluetooth Classic showcases an increase of 11.6% for both sending and receiving data, whilst BLE increases by 13.7% for sending data, however when receiving at the maximum rate, the noticed increase is only 3.4%. ESP-Now has the most drastic increase in power consumption at its highest transmission rate, with 25.8% for sending data and 11.2% for receiving.

Furthermore, BLE operates with much less power when sending and receiving data, regardless of its transmission rate, than any of the other standards tested, being, on average, 104.8% and 115.2% faster than Bluetooth Classic and 125.8% and 154.2% faster than ESP-Now. However, if the device is merely connected and is not trading information with another device, BLE consumes as much power as Bluetooth Classic version 4.2 and more than 100% less than ESP-Now.

Lastly, the variation of values that can be observed is not drastic on any transmission standard, however, for sole connection, it is higher on Bluetooth Classic. Other than that, it is almost non-existent barring some slight variation on some columns.

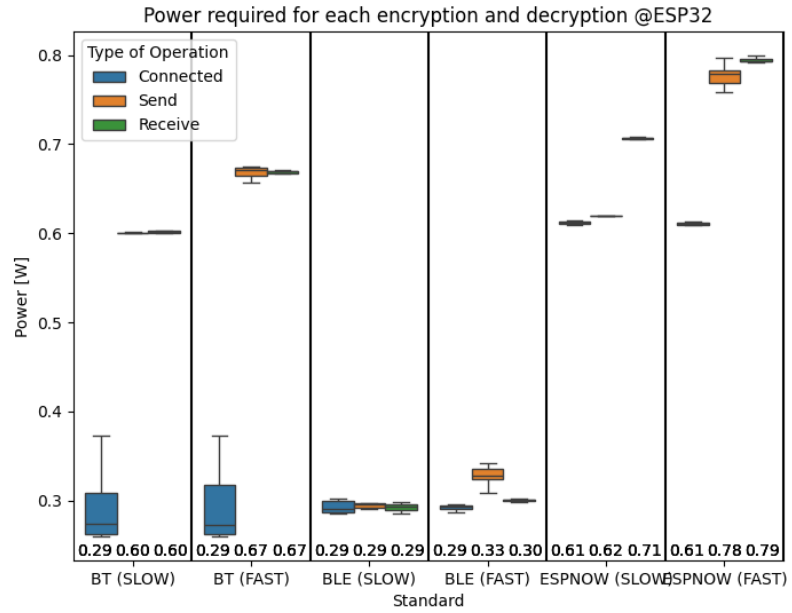


Figure 7.4: Power consumption of various standards in ESP32.

Figure 7.5 summarizes the results obtained for ESP8266. Given the lack of a Bluetooth module in this device, only ESP-Now results were assessed. When compared to ESP32, the variation in results is smaller, both between the slower and faster transmission modes, which represented a gain of 4.9% and 4.8% for sending and receiving data, respectively, and the transmission and reception of data on the same data rate. Power consumption was also less than ESP32, as it was expected given its lower computational capabilities.

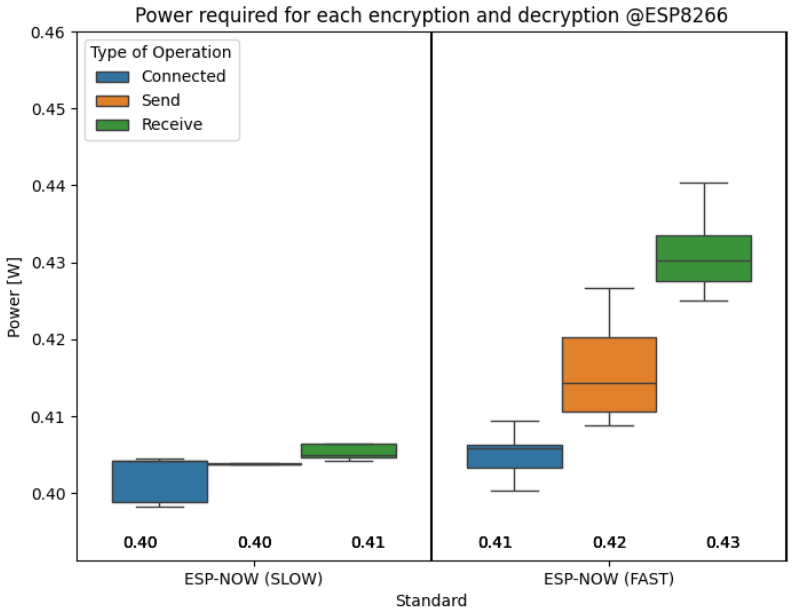


Figure 7.5: Power consumption of ESP-Now in ESP8266.

Continuing with the comparisons, Figure 7.6 compiles the Bluetooth Classic results for the Raspberry Pi Zero. The increase in power consumption between different transmission rates is consistent with the results obtained with the ESP32, with 12.6% and 13.7% increases for sending and receiving data, respectively, very similar to the previously mentioned results. As for the overall power consumption, the Pi Zero uses slightly more energy than the ESP32; this difference was expected to be larger given the big disparity in computational resources and clock speed between the two devices, similar to the gap observed between ESP32 and ESP8266.

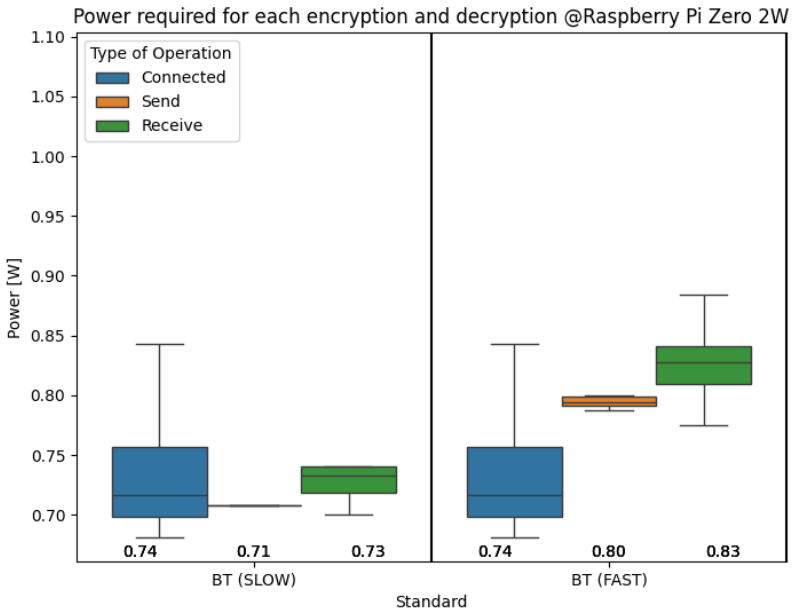


Figure 7.6: Power consumption of Bluetooth Classic v4.1 in Raspberry Pi Zero.

Lastly, in Figure 7.7, the power drawn by Bluetooth Classic on the Raspberry Pi 3B+ is showcased. The results are quite close to the Pi Zero when it comes to the comparisons between each standard and their transmission rate, with the faster version consuming 7.8% and 5.6% more energy for sending and receiving data, respectively, when compared to a slower one.

Beyond these discussed transmission standards, if looking for more accessibility, another interesting option could be Wi-Fi. The following Table 7.1 showcases the results for power consumption while using the device as both an Access Point and a Client, on the ESP32 and ESP8266. Furthermore, there are additional comparisons made in the power consumed by the usage of Wi-Fi and the previously discussed transmission standards. Overall, the results for the ESP32 are not very optimistic if looking for a power-saving option, with Wi-Fi using 160% more power than a device connected by Bluetooth and 23% more than ESP-Now. ESP8266 was only compared with ESP-Now, consuming 2.2% more energy, on average.

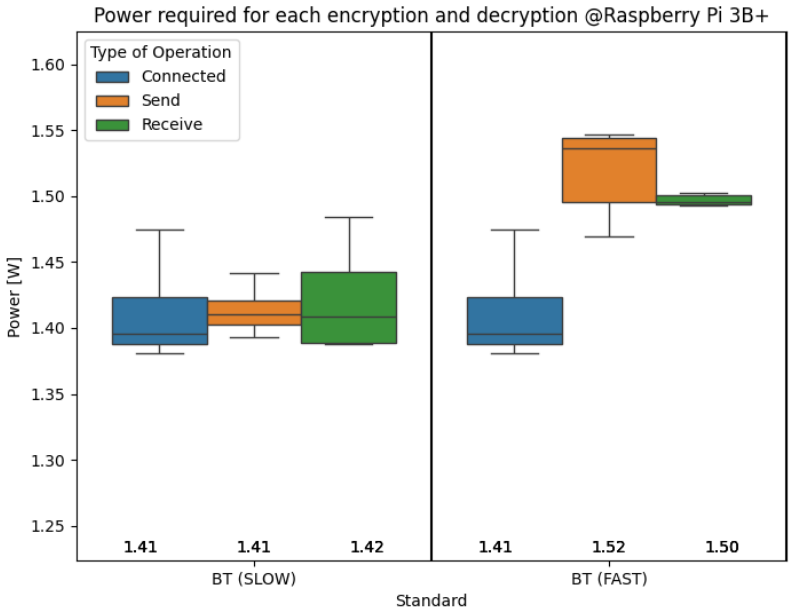


Figure 7.7: Power consumption of Bluetooth Classic v4.2 in Raspberry Pi 3B+.

	Access Point		Client	
	ESP32	ESP8266	ESP32	ESP8266
Power Measurements (W)	0.75	0.41	0.76	0.41
% difference to ESP-Now	+23.6%	+2.2%	+24.8%	+2.9%
% difference to BT	+160.1%	×	+162.5%	×
% difference to BLE	+160.1%	×	+162.5%	×

Table 7.1: Comparison of Wi-Fi against other standards.

One last important notion is how fast each standard is when transmitting data. Depending on the imposed scenario, a single device might need to transmit dozens, if not hundreds, of packets of information. This testing campaign included power measurements for a slow transmission rate and a fast transmission rate. In Table 7.2 the values for the slow and fast transmission are available.

	Slow Transmission			Fast Transmission		
	ESP-Now	BT	BLE	ESP-Now	BT	BLE
Packet Size (Bytes)	250	20	20	250	20	20
Packets per second	0.5	1	1	150	330	330
Bytes per second	125	20	20	37 500	6600	6600

Table 7.2: Different transmission rates for each standard.

7.3 Range Tests

This section includes all the data on the range tests conducted on each cipher to better understand their differences in range and compare the obtained values

with the theoretical ones provided in the background of this thesis, to see whether or not they correlate to the real-world results. Three tests were conducted in total, for the three different standards, all of them in an open and straight field, to get the most accurate and reliable assessment possible:

- **ESP-Now:** Both an ESP32 and an ESP8266 were used, with the latter staying fixed in position while the former was gradually carried away until a complete packet loss was noted.
- **Wi-Fi:** An ESP32 actively scanned for a mobile Access Point which was gradually carried away until complete packet loss.
- **BLE:** An ESP32 scanned for a defined Bluetooth client which was active and gradually carried away until full packet loss.

As shown in Figure 7.8, we can see a comparison in signal intensity across ESP-Now, Wi-Fi, and BLE and how it varies with an increasing distance. It is noticeable that Wi-Fi can still ensure a decent signal quality for much longer than the other two standards, staying above -70 dBm up to 25 meters of distance and transmitting reasonably good data up to 35 meters. Bluetooth on the other hand has a severely compromised range and it is noticeable with distances equal to or greater than 10 meters, which are already displaying values below the -80 dBm threshold.

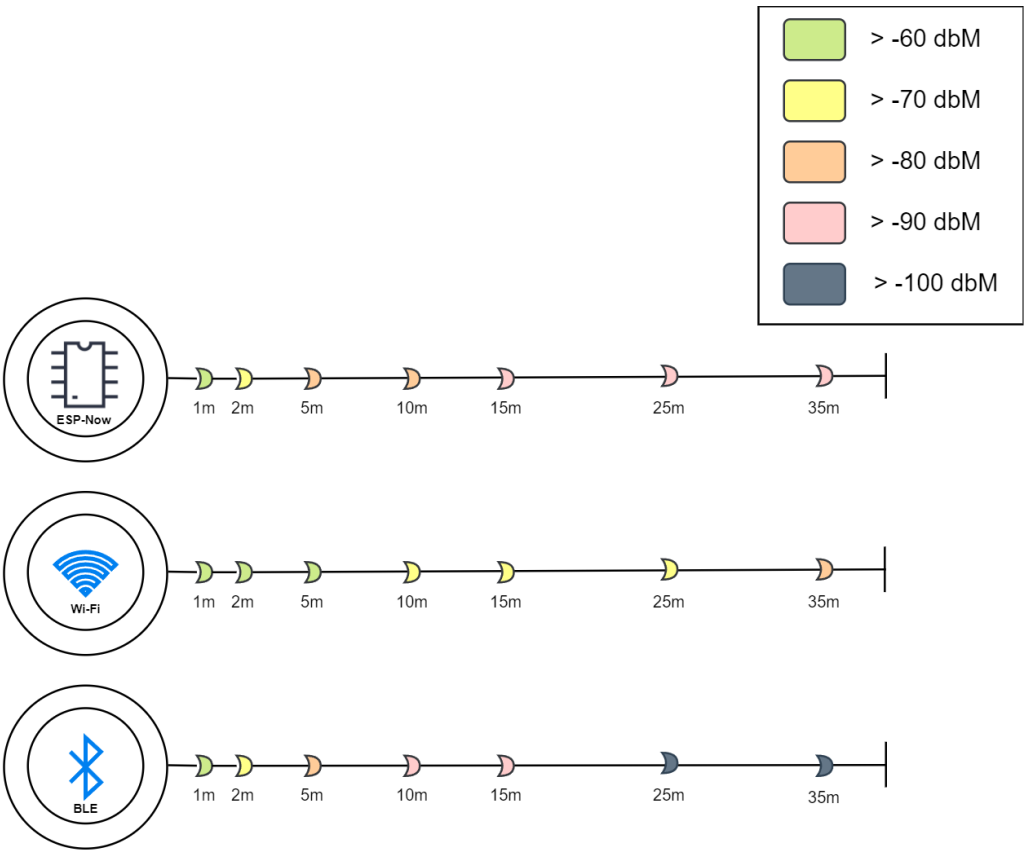


Figure 7.8: Range differential per standard.

Another important statistic regarding the range is the maximum distance each standard can cover whilst transmitting all of the data, half, or nearly zero. Table 7.3 has the raw information on which this sub-chapter was built, alongside three important statistics, namely, **"No Packet Loss"** which represents the maximum range found where no packets appeared to be lost, **"50% Packet Loss"**, which corresponds to the maximum distance where some packets would be lost in the transmission process, and finally **"100% Packet Loss"**, which represents the maximum range in which information occasionally would be delivered, but most of the time said data would be lost due to the distance being too big. As noticeable, Wi-Fi has by far the greater range in all three categories, being almost three times as effective for long-range communication than ESP-Now. BLE, as expected, operates with a much smaller range, which also plays a key part in providing such a low power consumption while transmitting data.

Standards	RSSI per Distance (dBm)							Maximum Range (m)		
	1 Meter	2 Meters	5 Meters	10 Meters	15 Meters	25 Meters	35 Meters	No Packet Loss	50% Packet Loss	100% Packet Loss
ESP-Now	−59	−66	−73	−76	−81	−83	−87	75	85	105
Wi-Fi	−45	−58	−59	−64	−66	−66	−73	200	225	285
BLE	−43	−65	−73	−81	−89	−93	−97	20	30	40

Table 7.3: Data gathered from range tests on the three standards.

7.4 Scenarios for their application

This section will provide further context into how the scenarios proposed in Section 6.5 could be supported by the various transmission standards analyzed. Once again, this decision will be pondered and made with the support of the data gathered in the previous sections of this chapter.

A1: Device to capture sea conditions

Transmission Standard chosen: Given their distance and to avoid or minimize the use of relay devices, Wi-Fi would probably be the best choice from the analyzed transmission standards purely on its range, however, it should be noted that if the distance is too big, other standards like LoRaWAN which have considerably more range than any of the studied ones might need to be used. If the usage of one or more relays would not be a problem then a power-saving standard would be preferred given their inaccessibility, thus, in this scenario, BLE would most likely be favored for its minimum consumption.

A2: Monitor whether a parking spot is empty or occupied

Transmission Standard chosen: As there is some concern with power saving, even though the devices are more accessible than in the previous scenario, BLE is once again the recommended choice for its lower power consumption.

A3: Monitor structural integrity of cranes

Transmission Standard chosen: Given its inaccessibility and the need to conserve as much battery as possible, the only logical choice should be, once again, the usage of BLE. Any other transmission standard is a synonym of higher power consumption, and thus not sustainable with the battery-constrained requirements imposed in this scenario.

A4: Capture data to support autonomous movement of cars in the seaport

Transmission Standard chosen: This scenario demands the maximum transmission rate and speed possible, given the need to support vehicle autonomy. On this factor alone, Wi-Fi should be the preferred choice for its much higher transmission rate. Even though its power consumption is marginally worse than ESP-Now and much worse than BLE, this alone does not warrant a switch for any of these transmission standards, given that they only support transmission rates much lower than the one offered by Wi-Fi, and thus not compatible with the requirements provided in this scenario.

A5: Receive data and create security backup

Transmission Standard chosen: The keywords to describe this scenario would be compatibility and range. Data will flow from multiple points and concentrate on this device, so a transmission standard with a higher range would likely be important to minimize the use of relays within the seaport as much as possible. Given this characteristic, both Wi-Fi and ESP-Now match the description, however, when it comes to compatibility, the latter will only work with ESP devices and, most likely, based on the complexity of the task at hand, a Raspberry Pi device would be the preferred choice, thus excluding this standard from consideration. Wi-Fi works on the larger portion of IoT hardware used, however, some doubts arise when it comes to its support of power-constrained devices that, more likely than not, will use BLE to save as much energy as possible in their batteries. A secondary approach to this problem could probably be to have intermediary devices that capture the data transmitted via BLE and relay to the higher levels of operation said data via Wi-Fi, thus eliminating any possible conflict previously held.

A6: High flow of data to coordinate seaport management

Transmission Standard chosen: Similarly to the previous scenario, compatibility is important to ensure data is transmitted between all necessary devices. Depending on whether several devices are used to control small networks of the seaport or one singular device receives the majority of the information to handle the data, the transmission standard to be chosen, as well as its range, will change. In the first case, BLE or even ESP-Now could be chosen given the low power consumption of the former and the compatibility and slightly increased range of the latter.

In the last case, Wi-Fi is the only logical choice given its vastly superior range, so that all information can reach the central point of this larger network.

The following Table 7.4 summarizes the presented scenarios to easily understand the best choices for each situation.

Scenarios	Transmission Standard			
	Wi-Fi	Bluetooth Classic	BLE	ESP-Now
A1	✓	×	✓	×
A2	×	×	✓	×
A3	×	×	✓	×
A4	✓	×	×	×
A5	✓	×	×	×
A6	✓	×	✓	✓

Table 7.4: Summary of transmission standards per scenario.

Chapter 8

Conclusion

8.1 Work Conducted

In this work, an extensive overview of lightweight encryption solutions for resource-constrained devices was provided, focusing on the seaport use case. Firstly, a background of the various concepts revolving around these topics was provided, namely specific encryption and hashing concepts, different lightweight ciphers and their characteristics, and the most used options for IoT devices and transmission standards, which are Arduino, Raspberry Pi, ESP hardware, and Wi-Fi, Bluetooth, and ESP-Now, respectively.

Furthermore, a literature review was carried out to gather current knowledge on the topics related to this study, with emphasis on the use of IoT technology in seaports, the available data on USB Power Meters as a tool to collect power-related metrics, and different studies on lightweight encryption, specifically pointing out how to compare encryption algorithms (implementations, the hardware used, software needed, etc...), as well as determine which metrics are commonly used to establish the cipher's performance and security analysis. Furthermore, a small study was also performed on quantum cryptography, which although is still far away from practical use cases that can compromise the security of some of the algorithms mentioned, represents a future threat to not only the lightweight environment, but the entirety of the encryption spectrum, with an exception given for some symmetric ciphers which have extensive key lengths of 256 bits or higher, as is the case of AES-256.

The methodology for the practical research was then established. Starting with an extensive search for publicly available and trustworthy implementations for the encryption algorithms mentioned previously, which yielded a small portion (about one-third) of ciphers that met this criterion. With that in mind, a security analysis based on existing information was carried out to find exactly the strong and weak points of each algorithm. This study was followed by research on the metrics to use, power-gathering devices, and a study of different scenarios in which to test our solutions, one inside the device with the data and three more using data transmission with Wi-Fi, Bluetooth, and ESP-Now.

Then, work was conducted to assess each cipher's and device's performance whilst encrypting and decrypting small packets of data (less than 64 Bytes). All the ciphers used for each device, namely AES, ASCON, TEA, and XTEA, are built on stable and reliable implementations, that have been tested outside of this thesis by multiple peers. One important note, however, is that ASCON was not implemented in both ESP devices due to constraints with the hardware itself that made running the selected implementations unfeasible. The findings were within the expected parameters defined in the methodology, with AES becoming increasingly less efficient both in speed and power consumption as its key size increases, whilst not yielding a big gain in security, given that 128-bits of key size, the minimum value for AES, is considered safe for the foreseeable future. Furthermore, in most of the devices studied, TEA and XTEA proved to be the more efficient ciphers, however, given some security concerns with the former, XTEA would be the only admissible one if robust security mechanisms were needed, although if the device and conditions are extremely resource-constrained, TEA could be considered given that, on average, yielded the most efficient performance out of all the ciphers. For most of the devices, ASCON was a comparable alternative to XTEA and more resource efficient than AES, proving that even with enhanced complexity, when compared to XTEA, can still provide similar results that adapt to the lightweight model in the study. As for the devices, Arduino Uno is by far the most resource-constrained device, followed by the Raspberry Pi Pico and ESP8266. ESP32 is then in a class of its own, far away from the two most demanding devices, the Pi 3B+ and Pi Zero. It is noteworthy to mention that with the increase in capabilities and speed efficiency, the power consumption also rises drastically, making such devices not suitable to perform low-level operations that require the usage of very low energy, such as battery-powered devices that are not easily reached or conducting work on them regularly is not financially viable. Another important aspect subject to analysis is the modes of operations chosen for each cipher, although ECB was used throughout most of the thesis, alternatives like GCM and CFB were also subject to analysis, both consuming more power and taking longer for both encryption and decryption when compared to ECB, although it is noteworthy to mention that they also provide a greater sense of security and even authentication in the case of GCM, which are two important features if dealing with IoT technology with vaster resources, like the higher-level scenarios A5 and A6 present in Section 6.5.

Further analysis was conducted to assess three transmission standards of interest for the IoT environment, namely Wi-Fi, Bluetooth, and ESP-Now. The work conducted points to a variant of Bluetooth focused on low energy usage, BLE, which is in a class of its own when it comes to power saving, consuming less than half of the next closest standard, Bluetooth Classic version 4.2. Another interesting point of analysis stems from their range, with Wi-Fi achieving higher distances than both Bluetooth and ESP-Now, thus proving its value when used to transmit data for a considerable range.

8.2 Future Work

The work on this thesis includes data collected from multiple lightweight ciphers and standards, which were applied to industry-used IoT devices, however, there is still some room to complement this analysis. Throughout the elaboration of the state-of-the-art, LoRaWAN was mentioned several times as a valuable standard to ensure data transmission with a range of several kilometers, something not achievable with the analyzed ones in this thesis. Although none of the used devices includes the LoRaWAN board necessary to use the standard, which was the key reason to not include it in this study, external hardware can be purchased and used to include LoRaWAN in the testing routine and, hopefully, offer an alternative to long-range scenarios, like A1 presented in Section 6.5, without the need for any data relaying and thus optimizing the overall structure of the IoT network in use.

Furthermore, another aspect that could improve the depth and quality of the work conducted on IoT lightweight technology would be the usage of hashing. Although this thesis contains some basic notions about it, there is no advanced research or comparison of any lightweight hashing functions, given that they were only considered as an option if using encryption was not viable in the context of resource-constrained devices. Future work could also analyze the performance of hashing functions and compare it with the results obtained using encryption, adapting it to some scenarios where confidentiality is not important, but data integrity is.

References

Omar A. Dawood and Othman I. Hammadi. An Analytical Study for Some Drawbacks and Weakness Points of the AES Cipher (Rijndael Algorithm). In *The 1 st International Conference on Information Technology (ICoIT'17)*, pages 129–133, Iraq, 4 2017. Lebanese French University. doi: 10.25212/icoit17.013.

Shapina Abdullah, ; Noorhayati, Mohamed Noor, ; Nor, Azimah Khalid, Zolidah Kasiran, Affiq Juzaily, Khairol Hisham, and Shah Alam. IOT Security: Data Encryption for Arduino-based IOT Devices. *Journal of Positive School Psychology*, 2022(3):8508–8516, 2022. URL `http://journalppw.com`.

Paul E A Adriaanse, Miray Ayşen, and Zekeriya Erkin. A Comparative Study of the TEA, XTEA, PRESENT and Simon lightweight cryptographic schemes. Technical report, Delft University of Technology, 7 2021.

Adrijana Agatić and Ines Kolanović. Improving the seaport service quality by implementing digital technologies, 2020. ISSN 13320718.

Abdullah Al-Mamun, Shawon S.M. Rahman, Tanvir Ahmed Shaon, and Md Alam Hossain. Security analysis of AES and enhancing its security by modifying s-box with an additional byte. *International Journal of Computer Networks and Communications*, 9(2):69–88, 3 2017. ISSN 09749322. doi: 10.5121/ijcnc.2017.9206.

Mahmood A. Al-Shareeda, Murtaja Ali Saare, Selvakumar Manickam, and Shankar Karuppayah. Bluetooth low energy for internet of things: review, challenges, and open issues. *Indonesian Journal of Electrical Engineering and Computer Science*, 31(2):1182–1189, 8 2023. ISSN 25024760. doi: 10.11591/ijeecs.v31.i2.pp1182-1189.

Abdullah Said Alkalbani, Teddy Mantoro, and Abu Osman Md Tap. Comparison between RSA hardware and software implementation for WSNs security schemes. In *Proceeding of the 3rd International Conference on Information and Communication Technology for the Moslem World: ICT Connecting Cultures, ICT4M 2010*, 2010. ISBN 9789791948913. doi: 10.1109/ICT4M.2010.5971920.

Hoda A Alkhzaimi and Martin M Lauridsen. Cryptanalysis of the SIMON Family of Block Ciphers. *IACR Cryptol. ePrint Arch.*, 2013, 2013. URL `https://api.semanticscholar.org/CorpusID:8595056`.

Subhadeep Banik, Andrey Bogdanov, and Francesco Regazzoni. Atomic-AES: A Compact Implementation of the AES Encryption/Decryption Core. In *International Conference in Cryptology, India*, 2016. doi: 10.1007/978-3-319-49890-4{_}10.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions. Technical report, 2011. URL http://sponge.noekeon.org/Version0.1.

Isha Bhardwaj, Ajay Kumar, and Manu Bansal. A Review on Lightweight Cryptography Algorithms for Data Security and Authentication in IoTs. *IEEE International Conference on Signal Processing, Computing and Control*, 4:504–509, 9 2017. doi: 10.1109/ISPCC.2017.8269731.

Alex Biryukov. Block Ciphers and Stream Ciphers: The State of the Art. Technical report, 2004. URL http://www.esat.kuleuven.ac.be/~abiryuko/.

Lilian Bossuet. Lightweight block ciphers implementations, 2016. URL https://perso.univ-st-etienne.fr/bl16388h/salware/lightweight_block_cipher.htm.

Huaifeng Chen and Xiaoyun Wang. Improved Linear Hull Attack on Round-Reduced Simon with Dynamic Key-guessing Techniques. Technical report, 3 2016.

Denhart. Tea Encryption on Arduino, 2010. URL https://github.com/dkobia/TEA-encryption-on-arduino.

Diy IOT. Guide to reduce the Arduino Power Consumption, 2021. URL https://diyi0t.com/arduino-reduce-power-consumption/.

Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *Journal of Cryptology*, 34(3), 7 2021. ISSN 14321378. doi: 10.1007/s00145-021-09398-9.

Mohammed El-hajj, Hussien Mousawi, and Ahmad Fadlallah. Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platform †. *Future Internet*, 15(2), 2 2023. ISSN 19995903. doi: 10.3390/fi15020054.

Espressif Systems. ESP-NOW Documentation, 2024. URL https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_now.html.

Andrew Froehlich. Wireless ad hoc network (WANET), 11 2022. URL https://www.techtarget.com/searchmobilecomputing/definition/ad-hoc-network.

Attlee Gamundani. An impact review on internet of things attacks. In *International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, pages 114–118, Windhoek, 5 2015. IEEE. ISBN 9781479977079. doi: 10.1109/ETNCC.2015.7184819.

P. Ganjar, W. Sakinah, R. Koekoeh, D. L. Dedi, and H. Indria. Realtime Measurement Integrated Hydrodynamic Conditions For Improving Port Performance. *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH*, 8(12):2375–2377, 12 2019. ISSN 2277-8616. URL `www.ijstr.org`.

Dharma Favitri Hariyanto, Iskandar, Ian Joseph Matheus Edward, and Tutun Juhana. Marine Radio for Voice Communication System on Very High Frequency (VHF) Spectrum . *IEEE 5th International Conference on Wireless and Telematics (ICWT)*, 2019. doi: 10.1109/ICWT47785.2019.8978220.

Julian Harttung. Mcrypton-vhdl, 2017. URL `https://github.com/huljar/mcrypton-vhdl`.

Julian Harttung. Prince-vhdl, 2021. URL `https://github.com/huljar/prince-vhdl`.

Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *Lecture Notes in Computer Science*, pages 46–59, Yokohama, 10 2006. CHES. doi: 10.1007/11894063{_}4.

Stephanie Hunn, Siti Zarina Naziri, and Norina Idris. The Development of Tiny Encryption Algorithm (TEA) Crypto-Core for Mobile Systems. In *2012 IEEE International Conference on Electronics Design, Systems and Applications (ICEDSA)*, pages 45–49, Kuala Lumpur, 11 2012. IEEE. ISBN 9781467321631. doi: 10.1109/ICEDSA.2012.6507813.

V. Hurbungs, T. P. Fowdur, and V. Bassoo. A power model for monitoring environmental parameters on the edge. In *International Conference on Electrical, Computer, Communications and Mechatronics Engineering, ICECCME 2021*, Mauritius, 10 2021. IEEE. ISBN 9781665412629. doi: 10.1109/ICECCME52200.2021.9590994.

IEEE Internet of Things Journal. IEEE Internet of Things Journal. URL `https://ieee-iotj.org/`.

Philipp Jovanovic. Piccolo, 2012. URL `https://github.com/Daeinar/piccolo`.

M Jović, E Tijan, S Aksentijević, and D Čišić. An Overview Of Security Challenges Of Seaport IoT Systems. In *42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1349–1354, Opatija, 5 2019. IEEE. doi: 10.23919/MIPRO.2019.8757206.

Gul N. Khan and Markus B. Moessner. Secure Authentication Protocol for RFID Systems. In *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–7, Lahaina, 7 2011. IEEE. ISBN 9781457706387. doi: 10.1109/ICCCN.2011.6006010.

George Klimiashvili, Cristiano Tapparello, and Wendi Heinzelman. LoRa vs. WiFi Ad Hoc: A Performance Analysis and Comparison. In *2020 International Conference on Computing, Networking and Communications (ICNC)*, pages 654–660, 2 2020. ISBN 9781728149059. doi: 10.1109/ICNC47757.2020.9049724.

Jia Hao Kong, Li Minn Ang, and Kah Phooi Seng. A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments, 2015. ISSN 10958592.

Tao Liu, Gowri Ramachandran, and Raja Jurdak. Post-Quantum Cryptography for Internet of Things: A Survey on Performance and Optimization. 1 2024. URL http://arxiv.org/abs/2401.17538.

Leo Louis. Working Principle of Arduino and Using it as a Tool for Study and Research. *International Journal of Control, Automation, Communication and Systems*, 1(2):21–29, 4 2016. ISSN 24557889. doi: 10.5121/ijcacs.2016.1203.

Calvin McCoy. Simon Speck Ciphers, 2018. URL https://github.com/inmcm/Simon_Speck_Ciphers.

David McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. Technical report, 2004.

Neil McNeight. Memory Free, 2016. URL https://github.com/McNeight/MemoryFree.

Thomas X Meng and William Buchanan. Lightweight Cryptographic Algorithms on Resource-Constrained Devices. 2020. doi: 10.20944/preprints202009.0302.v1. URL www.preprints.org.

Metageek. Understanding RSSI. URL https://www.metageek.com/training/resources/understanding-rssi/.

Dukjae Moon, Kyungdeok Hwang, Wonil Lee, Sangjin Lee, and Jongin Lim. Impossible differential cryptanalysis of reduced-round TEA and XTEA. In *Fast Software Encryption*, volume 2365, pages 49–60. IEEE, Rasht, 9 2002. ISBN 9783540440093. doi: 10.1007/3-540-45661-9{_}4.

Nicky Mouha. The Design Space of Lightweight Cryptography. *IACR Cryptology ePrint Archive*, 2015.

Anand Nayyar and Vikram Puri. Raspberry Pi-A Small, Powerful, Cost Effective and Efficient Form Factor Computer: A Review. *International Journal of Advanced Research in Computer Science and Software Engineering*, 5(12):720–737, 12 2015. ISSN 2277 128X. URL https://www.researchgate.net/publication/305668622.

Nick He. A Comprehensive Guide on Different Bluetooth Versions, 6 2024. URL https://www.mokosmart.com/guide-on-different-bluetooth-versions/.

NIST. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. Technical report, 2018. URL https://csrc.nist.gov/Projects/Lightweight-Cryptography.

Emmanuel Odunlade. Introduction to LoRa and LoRaWAN: What is LoRa and How Does It Work?, 5 2019. URL https://circuitdigest.com/article/introduction-to-lora-and-lorawan-what-is-lora-and-how-does-it-work.

Packetizer. AES Crypt Downloads. URL https://www.aescrypt.com/download/.

Pejman Panahi, Cüneyt Bayılmış, Unal Çavuşoğlu, and Sezgin Kaçar. Performance Evaluation of Lightweight Encryption Algorithms for IoT-Based Applications. *Arabian Journal for Science and Engineering*, 46(4):4015–4037, 4 2021. ISSN 21914281. doi: 10.1007/s13369-021-05358-4.

Pericle Perazzo, Francesca Righetti, Michele La Manna, and Carlo Vallati. Performance evaluation of Attribute-Based Encryption on constrained IoT devices. *Computer Communications*, 170:151–163, 3 2021. ISSN 1873703X. doi: 10.1016/j.comcom.2021.02.012.

Adithya Pokala. Piccolo Cipher, 2020. URL https://github.com/adipokala/piccolo-cipher.

Daniela Popescul. The Confidentiality-Integrity-Accessibility Triad into the Knowledge Security. A Reassessment from the Point of View of the Knowledge Contribution to Innovation. Technical report, 2011. URL https://www.researchgate.net/publication/257985911.

Michal Protasowicki. XTEA Cipher, 2021. URL https://reference.arduino.cc/reference/en/libraries/xtea-cipher/.

Muhammad Shahir Rahman, Sathwik Karnik, and Sumiyajav Sarangerel. Lightweight Cryptography. Technical report, 2022.

M Tolga Sakallı, Ercan Buluş, and Andaç Şahin. DIFFERENTIAL CRYPTANALYSIS FOR A 3-ROUND SPN. Technical report, 2005.

Federico Scarpa. Clefia, 2016. URL https://github.com/fedescarpa/clefia.

Frank Schmid. The Future of Cryptography and the Rise of Quantum Computing, 9 2023. URL https://www.genre.com/us/knowledge/publications/2023/september/the-future-of-cryptography-and-quantum-computing-en.

Anuj Sehgal, Vladislav Perelman, Siarhei Küryla, and Jurgen Schönwälder. Management of resource constrained devices in the internet of things. *IEEE Communications Magazine*, 50(12):144–149, 2012. ISSN 01636804. doi: 10.1109/MCOM.2012.6384464.

Je Sen. Slim-cipher, 2022. URL https://github.com/CryptoUSM/slim-cipher.

Jimil M. Shah, Roshan Anand, Satyam Saini, Rawhan Cyriac, Dereje Agonafer, Prabjit Singh, and Mike Kaler. Development of a technique to measure deliquescent relative humidity of particulate contaminants and determination of the operating relative humidity of a data center. In *ASME 2019 International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems, InterPACK 2019*. American Society of Mechanical Engineers (ASME), 2019. ISBN 9780791859322. doi: 10.1115/IPACK2019-6601.

Sony. The 128-bit Blockcipher CLEFIA Design Rationale Revision. Technical report, 2007.

Matej Sychra. AESLib, 2023. URL `https://www.arduino.cc/reference/en/libraries/aeslib/`.

Baraa Tareq Hammad, Norziana Jamil, Mohd Ezanee Rusli, and Muhammad Z Reza. A survey of Lightweight Cryptographic Hash Function. *International Journal of Scientific & Engineering Research*, 8(7), 2017. ISSN 2229-5518. URL `http://www.ijser.org`.

Harshvardhan Tiwari. Merkle-Damgård Construction Method and Alternatives: A Review. *Journal of Information and Organizational Sciences (JIOS)*, 41(2):283–304, 12 2017. doi: 10.31341/jios.41.2.9.

Dnislam Urazayev, Aida Eduard, Dimitrios Zorbas, and Muhammad Ahsan. Indoor Performance Evaluation of ESP-NOW. Technical report, 2023. URL `https://github.com/espressif/esp-now`.

Alexander Victor, Tietgen Bardram, Delbo Larsen, Mateusz Malarski, Martin Nordal Petersen, and Sarah Ruepp. LoRaWan Capacity Simulation and Field Test in a Harbour Environment. Technical report, 2018.

Deepali Virmani, Nidhi Beniwal, Gargi Mandal, and Saloni Talwar. Enhanced Tiny Encryption Algorithm with Embedding (ETEA). *International Journal of Computers & Technology*, 7(1), 2013. doi: 10.24297/ijct.v7i1.3479.

Jeffrey Walton. Cryptopp, 2023. URL `https://github.com/weidai11/cryptopp/tree/master`.

Rhys Weatherley. Arduino Cryptography Library, 2023a. URL `https://rweather.github.io/arduinolibs/crypto.html`.

Rhys Weatherley. Ascon-Suite, 2023b. URL `https://github.com/rweather/ascon-suite`.

Timberwolf White. Block cipher mode of operation.

Herbert Xu. Tea Cipher, 2022a. URL `https://github.com/torvalds/linux/blob/master/crypto/tea.c`.

Jinyan Xu. Qarma64, 2022b. URL `https://github.com/Phantom1003/QARMA64`.

Yongsheng Yang, Meisu Zhong, Haiqing Yao, Fang Yu, Xiuwen Fu, and Octavian Postolache. Internet of things for smart ports: Technologies and challenges. *IEEE Instrumentation and Measurement Magazine*, 21(1):34–43, 2 2018. ISSN 10946969. doi: 10.1109/MIM.2018.8278808.

Ibrar Yaqoob, Ejaz Ahmed, Muhammad Habib ur Rehman, Abdelmuttlib Ibrahim Abdalla Ahmed, Mohammed Ali Al-garadi, Muhammad Imran, and Mohsen Guizani. The rise of ransomware and emerging security challenges in the Internet of Things. *Computer Networks*, 129:444–458, 12 2017. ISSN 13891286. doi: 10.1016/j.comnet.2017.09.003.

Lei Zhang and Wenling Wu. LBlock: A Lightweight Block Cipher. In *Applied Cryptography and Network Security - 9th International Conference*, pages 327–344, Nerja, 6 2011. ACNS. doi: 10.1007/978-3-642-21554-4{_}19. URL `https://www.researchgate.net/publication/221651896`.

Appendices

Appendix A

Metrics for the ciphers

The first portion of this appendix will cover a summary of all results gathered from the practical experiments conducted on encryption and decryption of the selected ciphers on the chosen IoT devices. Table A.1 contains all the data for encryption exclusively separated by the device. Since there was no analysis conducted on ASCON for both ESP devices, these entries are blank.

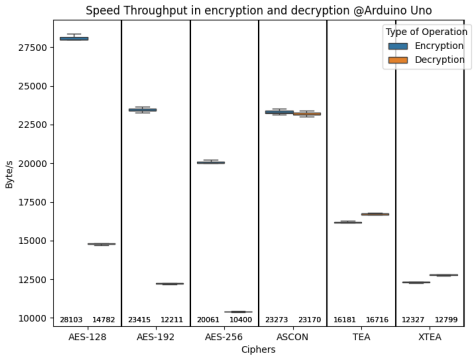
		Metrics						
		Time [µs]	**Time Standard Deviation**	**Speed Throughput [Bytes/s]**	**µs/Byte**	**Speed Latency [Cycles/block]**	**Power Measurement [W]**	**Power Latency [µJ/bit]**
Arduino	AES-128	569	3.1809	28103	35.58	9110	0.119	0.5262
	AES-192	683	3.5114	23415	42.71	10934	0.1189	0.6317
	AES-256	798	3.1415	20061	49.85	12761	0.1178	0.7307
	ASCON	688	3.3764	23273	42.97	11000	0.125	0.6689
	TEA	989	2.2616	16181	61.80	15822	0.1205	0.9301
	XTEA	1298	2.8810	12327	81.12	20767	0.1183	1.2162
ESP8266	AES-128	96	0.5424	166640	6	7681	0,1421	0.632
	AES-192	115	0.4421	139365	7.18	9185	0.1316	0.703
	AES-256	134	0.4476	119583	8.36	10704	0.1381	0.861
	ASCON	-	-	-	-	-	-	-
	TEA	22	1.0949	720268	1.39	1780	0.1399	1.081
	XTEA	28	3.9221	575135	1.75	2237	0.1391	1.075
ESP32	AES-128	18	0.2683	889778	1.12	4317	0.340	1.511
	AES-192	19	0.1413	842515	1.19	4558	0.338	1.806
	AES-256	19	0.0894	842303	1.19	4559	0.340	2.117
	ASCON	-	-	-	-	-	-	-
	TEA	10	0.2683	1598800	0.63	2403	0.352	2.720
	XTEA	13	1.1180	1229150	0.82	3132	0.359	2.774
Pico	AES-128	56	2.5977	285182	3.51	7026	0.116	0.516
	AES-192	66	2.6502	242956	4.12	8244	0.117	0.623
	AES-256	76	3.0095	210043	4.77	9536	0.117	0.732
	ASCON	40	1.8986	405146	2.47	4946	0.121	0.651
	TEA	22	1.9644	743545	1.35	2706	0.119	0.919
	XTEA	35	2.9674	457940	2.19	4390	0.121	1.085
Zero	AES-128	0.029	0.0069	5.8e+08	0.0018	29	1.159	5.154
	AES-192	0.035	0.0075	4.8e+08	0.0022	35	1.123	5.997
	AES-256	0.040	0.0089	4.1e+08	0.0025	40	1.162	7.240
	ASCON	0.015	0.0084	1.3e+09	0.0009	15	1.240	6.662
	TEA	0.011	0.0060	1.6e+09	0.0007	11	1.011	7.808
	XTEA	0.013	0.0025	1.3e+09	0.0008	13	1.013	9.051
3B+	AES-128	0.034	0.0122	5.0e+08	0.0021	47	1.598	7.106
	AES-192	0.036	0.0051	4.5e+08	0.0023	51	1.610	8.597
	AES-256	0.040	0.0050	4.1e+08	0.0025	55	1.610	10.030
	ASCON	0.018	0.0097	1.1e+09	0.0011	25	1.616	8.678
	TEA	0.011	0.0007	1.4e+09	0.0007	16	1.536	11.866
	XTEA	0.014	0.0018	1.1e+09	0.0009	20	1.558	13.916

Table A.1: Encryption metrics collected throughout the testing phase

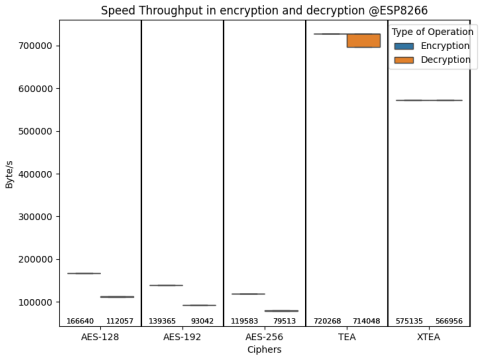
Table A.2 contains all the data for decryption exclusively separated by the device. Once again, since there was no analysis conducted on ASCON for both ESP devices, these entries are blank.

		Metrics						
		Time [µs]	**Time Standard Deviation**	**Speed Throughput [Bytes/s]**	**µs/Byte**	**Speed Latency [Cycles/block]**	**Power Measurement [W]**	**Power Latency [µJ/bit]**
Arduino	AES-128	1082	2.2733	14782	67.65	17319	0.117	0.991
	AES-192	1310	3.1704	12211	81.90	20965	0.117	1.201
	AES-256	1539	3.6873	10400	96.16	24617	0.117	1.406
	ASCON	691	3.3338	23170	43.16	11049	0.125	0.675
	TEA	957	2.3926	16716	59.82	15314	0.121	0.902
	XTEA	1250	2.8684	12799	78.13	20002	0.112	1.034
ESP8266	AES-128	143	2.1632	112057	8.93	11425	0.135	1.365
	AES-192	172	2.1211	93042	10.75	13759	0.141	1.193
	AES-256	201	2.2263	79513	12.58	16100	0.137	1.403
	ASCON	-	-	-	-	-	-	-
	TEA	22	0.7138	714048	1.40	1794	0.139	0.753
	XTEA	28	1.3057	566956	1.77	2260	0.140	0.756
ESP32	AES-128	13	0.0894	1230441	0.81	3121	0.338	3.423
	AES-192	14	0.0447	1142705	0.88	3360	0.327	2.767
	AES-256	14	0.0894	1142571	0.88	3361	0.339	3.472
	ASCON	-	-	-	-	-	-	-
	TEA	10	0.0894	1599467	0.63	2401	0.358	1.933
	XTEA	13	0.4472	1229699	0.81	3125	0.360	1.944
Pico	AES-128	92	3.6493	174210	5.75	11496	0.118	1.000
	AES-192	111	3.3822	144733	6.92	13831	0.118	1.212
	AES-256	130	4.0434	122843	8.15	16296	0.117	1.412
	ASCON	40	2.3089	405809	2.47	4942	0.121	0.650
	TEA	22	1.7989	742026	1.35	2708	0.123	0.923
	XTEA	29	2.1742	546471	1.84	3674	0.121	1.042
Zero	AES-128	0.0101	0.0069	5.4e+08	0.0019	31	1.082	9.148
	AES-192	0.036	0.0078	4.7e+08	0.0022	36	1.122	11.484
	AES-256	0.041	0.0084	4.1e+08	0.0026	41	1.191	14.311
	ASCON	0.013	0.0054	1.3e+09	0.0008	13	1.180	6.364
	TEA	0.011	0.0015	1.5+09	0.0007	11	1.017	7.603
	XTEA	0.013	0.0027	1.2e+09	0.0008	11	0.989	8.524
3B+	AES-128	0.032	0.0052	5.1e+08	0.0020	45	1.586	13.413
	AES-192	0.038	0.0077	4.3e+08	0.0024	54	1.583	16.206
	AES-256	0.042	0.0052	3.9e+08	0.0026	58	1.579	18.980
	ASCON	0.015	0.0058	1.2e+09	0.0009	21	1.617	8.723
	TEA	0.011	0.0006	1.4e+09	0.0007	16	1.527	11.419
	XTEA	0.014	0.0040	1.1e+09	0.0009	20	1.533	13.219

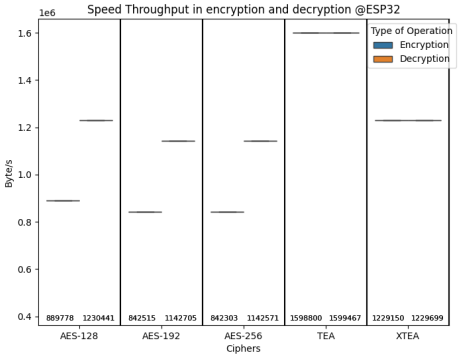
Table A.2: Decryption metrics collected throughout the testing phase



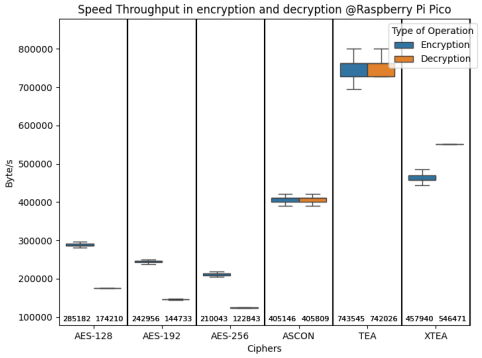
(a) Arduino Uno



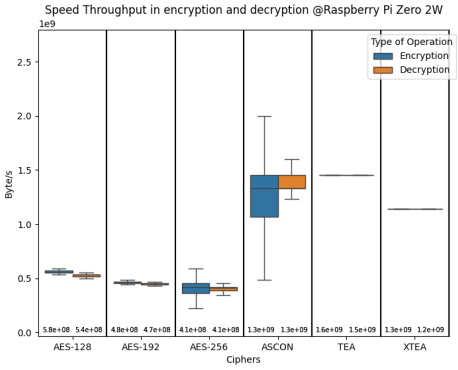
(b) ESP8266



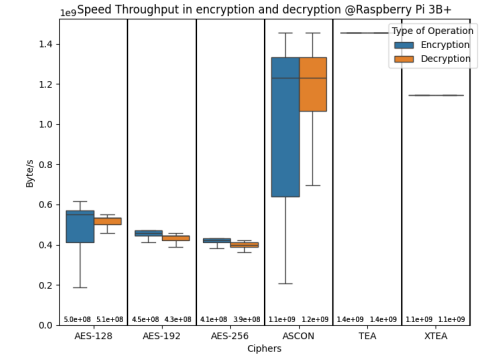
(c) ESP32



(d) Raspberry Pi Pico

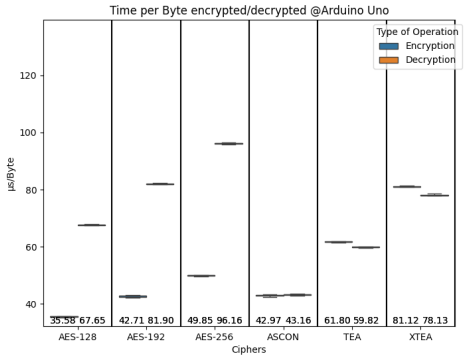


(e) Raspberry Pi Zero

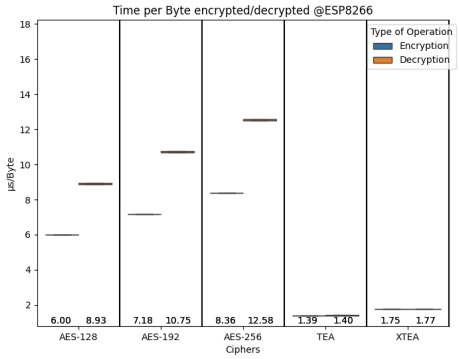


(f) Raspberry Pi 3B+

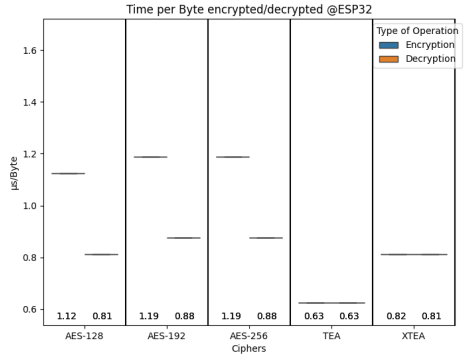
Figure A.1: Speed throughput for all devices upon encryption and decryption.



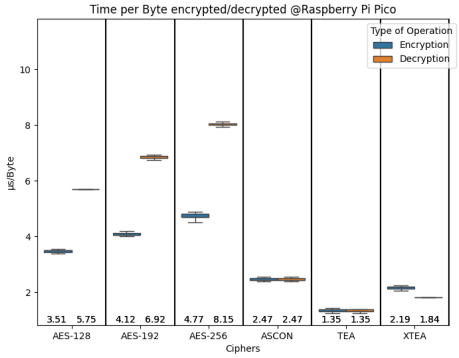
(a) Arduino Uno



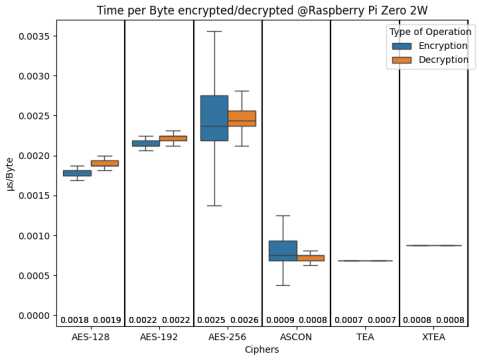
(b) ESP8266



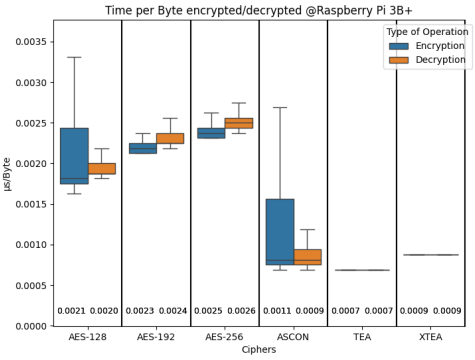
(c) ESP32



(d) Raspberry Pi Pico

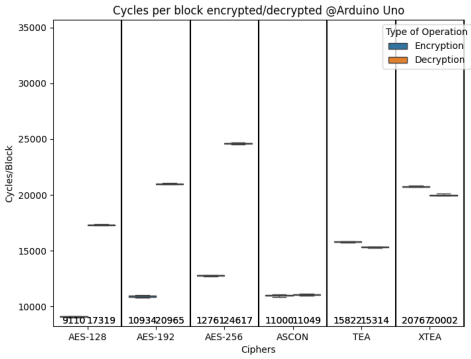


(e) Raspberry Pi Zero

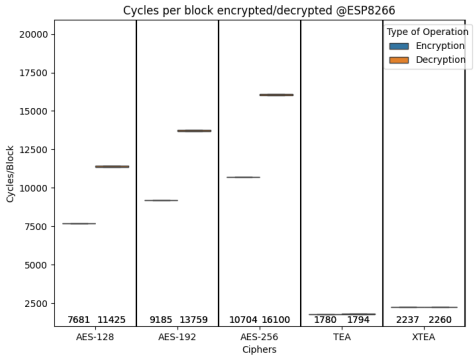


(f) Raspberry Pi 3B+

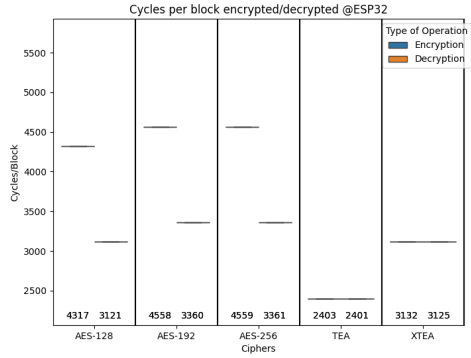
Figure A.2: Time per byte encrypted and decryption on all devices.



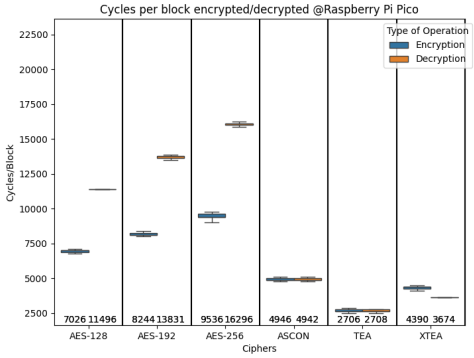
(a) Arduino Uno



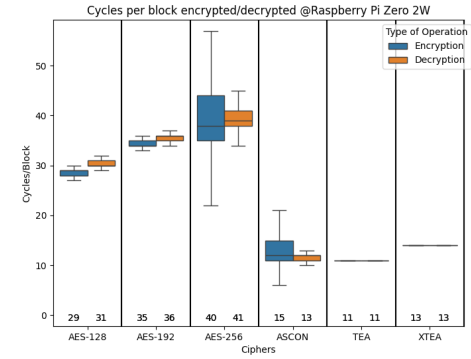
(b) ESP8266



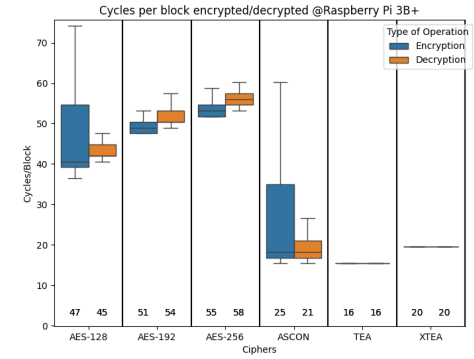
(c) ESP32



(d) Raspberry Pi Pico



(e) Raspberry Pi Zero



(f) Raspberry Pi 3B+

Figure A.3: Speed latency on all devices.