technologies

Article

A Survey of Efficient Lightweight Cryptography for Power-Constrained Microcontrollers

Jesús Soto-Cruz [1], Erica Ruiz-Ibarra [1,*], Javier Vázquez-Castillo [2], Adolfo Espinoza-Ruiz [1], Alejandro Castillo-Atoche [3] and Joaquin Mass-Sanchez [1]

[1] Departamento de Ingeniería Eléctrica y Electrónica, Instituto Tecnológico de Sonora, ITSON, Ciudad Obregón 85130, Mexico; jesusa.soto@potros.itson.edu.mx (J.S.-C.); adolfo.espinoza@itson.edu.mx (A.E.-R.); joaquin.mass@itson.edu.mx (J.M.-S.)
[2] Departamento de Redes e Informática, Universidad Autónoma del Estado de Quintana Roo (UQRoo), Chetumal 77019, Mexico; jvazquez@uqroo.edu.mx
[3] Departamento de Mecatrónica, Universidad Autónoma de Yucatán, Mérida 97000, Mexico; acastill@correo.uady.mx
* Correspondence: erica.ruiz@itson.edu.mx

Abstract: Protecting sensitive data, such as data collected from sensors, is crucial for ensuring the accurate assessment of sensing devices and preventing unauthorized access. In this regard, Internet of Things (IoT) devices offer a promising alternative for in situ monitoring. However, IoT sensing devices are often constrained by limited processing power and memory. Therefore, lightweight and efficient security algorithms are essential. This review paper assesses the implementation of lightweight cryptographic algorithms for power-constrained microcontrollers in IoT systems using symmetric cryptography. The implemented algorithms analyze the resource constraints of the IoT devices and compare their performance for the efficient implementation of secure monitoring systems. Experimental results demonstrate the performance of various lightweight encryption algorithms on low-power microcontrollers. The analysis reveals the comparative performance of these algorithms in terms of average power and energy consumption, memory usage, latency, and throughput.

Keywords: Internet of Things; data security; wireless sensor network; lightweight encryption

check for updates

Academic Editors: Esther Salmerón-Manzano and Francisco Manzano Agugliaro

Received: 29 October 2024
Revised: 29 November 2024
Accepted: 5 December 2024
Published: 24 December 2024

Citation: Soto-Cruz, J.; Ruiz-Ibarra, E.; Vázquez-Castillo, J.; Espinoza-Ruiz, A.; Castillo-Atoche, A.; Mass-Sanchez, J. A Survey of Efficient Lightweight Cryptography for Power-Constrained Microcontrollers. *Technologies* **2025**, *13*, 3. https://doi.org/10.3390/technologies13010003

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Wireless sensor networks (WSNs) play a crucial role in extending the Internet's reach to a diverse range of devices and environments [1–5]. These networks comprise numerous small, low-power devices equipped with sensors to gather data from their surroundings. WSNs are widely employed in various applications, such as vehicular networks [6], Internet of Medical Things [7], industrial automation [8], environmental monitoring [9], and healthcare [10]. Even though WSNs offer great versatility and are great options for implementing applications, these networks are susceptible to security vulnerabilities due to their open wireless nature and limited device resources.

WSNs and Internet of Things (IoT) devices face several performance challenges and energy constraints that are critical issues impacting their functionality and efficiency; as the implementation of these devices in several applications is increasing over time, understanding their limitations is essential for optimizing their operations. IoT devices rely on batteries or energy-harvesting methods, resulting in restrictions over operational time and processing capabilities [11]; also, tasks such as data processing, security measures, and

communication protocols introduce additional computational and energy requirements on resource-constrained devices [12,13]. On top of that, effective communication protocols are required in WSNs to ensure that data are transmitted reliably without excessive energy use, hindering overall network performance [14]. In addition to this, IoT and WSN devices typically have low processing power, making it difficult to efficiently execute complex algorithms or process large amounts of data [14].

Safeguarding sensitive data within WSNs is a paramount challenge. Additionally, ensuring secure integration, preserving privacy, and mitigating risks associated with the vast amount of data transmitted across wireless devices are other critical WSN security concerns [15]. Cyberattacks targeting IoT devices can have severe consequences, including data breaches, network congestion, and physical damage to devices. There are mechanisms designed to minimize the vulnerabilities of IoT systems to avoid cyberattacks; unfortunately, traditional security mechanisms are not well suited for IoT due to its interconnected nature and the limited resources of IoT devices [16,17]. Figure 1 shows the security mechanisms used in IoT to counter attacks targeting the network or its devices; one of the most relevant and crucial mechanisms in the context of IoT is cryptography, as it facilitates secure communication between devices, in addition to data protection and user privacy [18]. Despite this, traditional cryptography is not a viable mechanism for IoT devices as it is highly resource demanding [19]. As an alternative to traditional cryptography, and considering the resource-constraints of these devices, a new branch of cryptography, known as lightweight cryptography, has been introduced and designed specifically for this type of environment [20]. Lightweight cryptography is a rapidly evolving field aimed at providing security for devices with resource constraints, such as those used for IoT and WSNs. For this reason, recent advances in this field focus on the development of novel approaches by improving the performance of existing algorithms, either by optimizing its implementations or by improving their design [21–23].

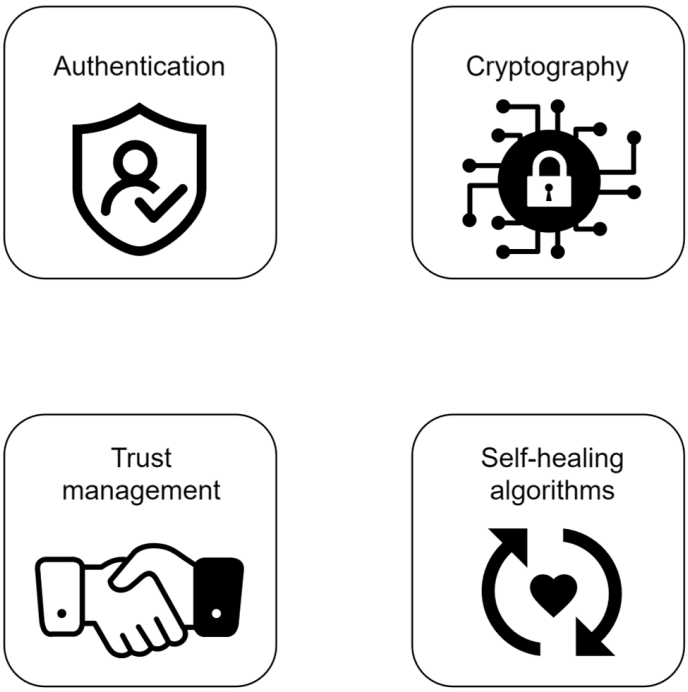


Figure 1. Security mechanisms used in IoT.

The feasibility of using lightweight cryptography to secure IoT devices with limited power and computational resources has recently started being investigated. Lightweight cryptography is developed to efficiently implement algorithms on resource-constrained

devices. This technique optimizes hardware and software implementations, minimizing power consumption, memory footprint, and computational overhead. In conventional cybersecurity implementations, symmetric and asymmetric techniques are used to maximize the security of applications, with each approach offering distinct advantages and limitations. However, symmetric cryptography is generally the preferred choice for low-power consumption implementations, especially in resource-constrained devices like IoT devices. For this reason, this survey focuses on symmetric cryptography, where ten cryptography algorithms are evaluated, comparing the algorithms' resource usage and their ability to ensure secure monitoring applications. This is performed through the implementation of relevant lightweight cryptographic algorithms for power-constrained devices reviewed in the state of the art for IoT security. The implemented algorithms are analyzed and compared for the efficient implementation of secure monitoring systems. The experimental results obtained as a product of this work are of great value for determining the performance of the different lightweight algorithms reviewed and for determining their best use on low-power microcontrollers, depending of their application.

The following sections are organized as follows: Section 2 shows related works about the use of cryptographic algorithms and their performance. Section 3 describes the lightweight ciphers implemented in this work. In Section 4, the methodology of this work and the performance metrics used to evaluate the lightweight cipher algorithms are described, in addition to the development boards where the lightweight ciphers were implemented, whose results are shown and discussed in Section 5. Section 6 discusses the findings in this work and outlines the integration of symmetric algorithms with more complex systems. Finally, the conclusions for this work are presented in Section 7.

2. Related Work

Within cryptography itself, there are two types of methods used to protect transmitted information: symmetric and asymmetric encryption [24]. In the former, a single key is used to encrypt and decrypt messages, analogous to how a key is used to open and close a lock. In the latter, a pair of keys (a public and a private key) are used, where a key is used to encrypt a message that can only be decrypted using the other key. Asymmetric encryption in comparison with symmetric encryption requires significant computational power and memory, making them less suitable for resource-constrained devices [25,26]. Also, asymmetric encryption is generally slower than symmetric methods; in comparison with the latter, there are only a few asymmetric algorithms accessible for this kind of cipher [27]. For this reason, the work presented in this paper focuses only on the first method, symmetric encryption; thus, more will be said about this type of algorithms. The operation of these algorithms is easy to understand at a large scale; Figure 2 displays the operation of this cipher. The message M can be encrypted with the AES function as the encryption algorithm in combination with the key K_1, producing an encrypted message EM. If the inverse AES^{-1} is performed using the same key and the encrypted message EM, a message M' is produced, identical to M.

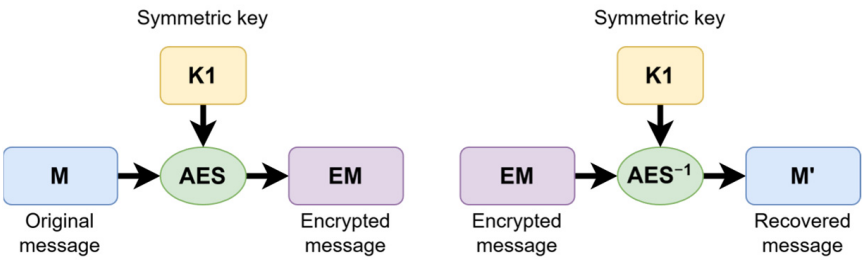


Figure 2. Symmetric encryption process diagram.

There are two types of symmetric encryption algorithms: block cipher and stream cipher. They both fulfill the function of encrypting or decrypting a message; the main difference between these two is that the block cipher works with blocks of fixed size, in which operations are performed using bits of the message and the key [28], whereas the stream cipher generates a pseudo-random vector called "stream key" and encrypts the message with it using an XOR operation [29]. Figure 3 describes the encryption and decryption process used in a block cipher, and Figure 4 illustrates that of a stream cipher. Both ciphers use the same process shown in Figure 2; the main difference between them is that in a block cipher, a function is needed to encrypt and the inverse of that function is needed to decrypt, but in a stream cipher, the same function is used for encryption as well for decryption, as the inverse XOR operation is the same operation.

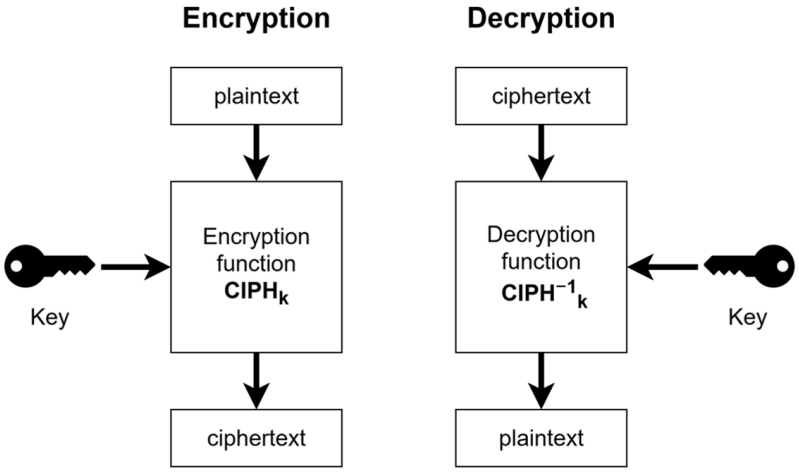


Figure 3. Encryption and decryption process using a block cipher algorithm.

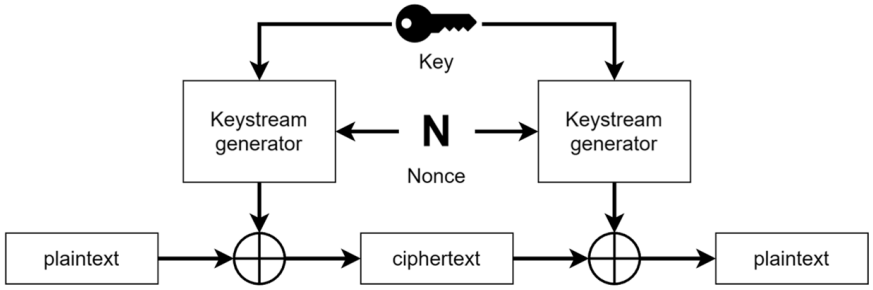


Figure 4. Encryption and decryption process using a stream cipher algorithm.

Symmetric cryptography has recently started to be studied for its use in IoT; Hatzivasilis et al. [30] conducted a survey on 52 lightweight block ciphers both in software and hardware, all of which were evaluated according to their security (as shown in the reviewed state of the art) and performance in terms of memory consumption, energy, and implementation complexity. This survey highlights vulnerabilities in the algorithms and emphasizes the need for further analysis of the new lightweight algorithm proposals. Similarly, Jassim S. and Farhan A. [31] conducted a survey focused on 27 lightweight stream ciphers, evaluating their design patterns, key size, vulnerabilities shown in previously performed studies, and adaptability for constrained devices. Both Hatzivasilis et al. [30] and Jassim S. and Farhan A. [31] highlight the necessity of lightweight cryptography for IoT to ensure security; however, these surveys do not specifically mention the devices used to implement the algorithms for its evaluation, causing uncertainty for developers wanting to implement such solutions on devices of their preference.

T. Meng et al. [32] implemented and evaluated both block and stream cipher algorithms, focusing mostly on the amount of memory and CPU usage required to execute the algorithms, along with the execution time and throughput for encryption and decryption. This paper does not evaluate a vast number of algorithms like the previous surveys, but it identifies performance metrics for the ciphers used and compares them with each other. Unfortunately, the implementation of the studied algorithms is performed on devices with a substantially high quantity of resources; for this reason, the comparison between the results obtained in this paper with studies using devices with constrained resources is not ideal.

R. Kureshi and B. Mishra [33] performed an evaluation of block cipher algorithms on a Raspberry Pi 3B+ development board. It should be noted that the algorithms used in this study are not lightweight, and the board itself has a large amount of resource compared with those commonly used in IoT; therefore, this study focuses on traditional encryption algorithms for IoT.

A. Sevin and A. Osman [34] presented a study of 50 lightweight block ciphers implemented in software for IoT to evaluate their performance with respect to execution time, latency, ROM and RAM usage, and energy. The algorithms presented are implemented on an 8-bit ATmega128 microcontroller with low power consumption, commonly used in IoT applications. This study aids IoT developers regarding the selection of an appropriate block cipher algorithm for the requirements of their specific applications and platforms. In addition to this study, the authors suggest that these algorithms studied should be used and tested on different platforms such as development boards with 16-bit and 32-bit architectures, as well as pointing out the need to evaluate stream ciphers.

L. Jiao et al. [35] reviewed the history of stream ciphers and how they have been developed over the years in such a way that these ciphers can be efficient on hardware implementations without neglecting an adequate level of security, even with resource constraints. This review classified stream ciphers into groups according to their design principles, then used this groups to discuss their strengths and weaknesses as well as design proposals to improve their performance. A theoretical and analytical approach to the compared stream ciphers rather than implementing them is taken in this paper, helping to understand the operation of such ciphers in order to establish a basis for future designs. This paper is not suitable for the performance evaluation of the algorithms presented as no variables of interest are shown that can be useful to developers looking for specific needs.

V. Thakor et al. [36], similarly to [30], conducted a survey of 50 popular block ciphers used in IoT devices, evaluating them regarding their design features, performance, and security measures. Security issues were also addressed, taking into consideration the constrained resources of IoT devices, stressing the importance of lightweight cryptography for secure data transfer. The algorithms shown were implemented on microcontrollers from the NXP, AVR, and ARM platforms. The algorithms were evaluated based on their structure design, assisting their comparison and helping in the decision making of the developers, depending on their needs. Despite briefly mentioning stream cipher algorithms, this paper does not evaluate algorithms of this type, as the authors consider that block ciphers are preferred over stream ciphers due to the security level they provide along with their versatility and simplicity of design.

The previously mentioned papers are valuable contributions providing an overview of the options available when looking for a suitable symmetric encryption algorithm to be used on IoT applications with specific needs. However, the field still needs to be thoroughly explored in order to provide more information and have a wider scope of all available options. One of the branches that should be further studied involves stream ciphers, given

that they are a good alternative for data protection and that most of the papers mentioned in this section does not include them.

3. Symmetric Cipher

This section describes the algorithms evaluated in the development of this work; these are organized according to the type of symmetric cipher they belong to: block cipher or stream cipher.

Symmetric ciphers have a crucial role in the IoT environment, ensuring the security of information between devices and their data [32]. Traditional encryption algorithms used in computers, such as the AES standard, are a valuable tool used to protect information; however, they require a considerable amount of memory and processing power, affecting the performance of IoT devices as well as their lifetime [37]. Therefore, it is highly advisable to use lightweight encryption algorithms that use the least quantity of computational resources possible and guarantee a strong level security as a security mechanism for IoT.

The selection of cryptographic algorithms was performed, taking into account the wide diversity of design architectures in order to provide a wide range of use cases and possible threats as well as determining which type of algorithm is most suited depending on the selected devices. Also, recognized algorithms in the cryptographic community were considered. Ultimately, only algorithms within the "lightweight" category were accounted for.

The following algorithms were evaluated in this study.

3.1. Lightweight Block Ciphers

- PRESENT [38]: This algorithm is designed for environments with limited resources, such as RFID cards and small embedded systems. It uses a 64-bit data block and an 80-bit or 128-bit key. Its design is optimized for hardware implementation, with a major focus on low power consumption. Tests have shown that it is resistant to brute force attacks, although it was also shown that the key used for encryption can be recovered by differential fault analysis attacks [39].
- SPECK [40]: Developed by the National Security Agency (NSA), this cipher was designed for efficient performance in resource-constrained environments, and it employs different blocks sizes as well as key sizes. It has been revealed that this cipher has vulnerabilities when subjected to different cryptoanalysis attacks, especially in scenarios where a small number of rounds are used for encryption [41]. Despite this, the algorithm is considered a great alternative due to its high performance in IoT applications.
- Piccolo [42]: It is an ultra-lightweight algorithm that operates with a block size of 64 bits and an 80-bit or 128-bit key size designed to be efficient in hardware implementations. Several vulnerabilities have been found when performing different crypto-analytic attacks [43,44]. Despite the fact that this cipher is efficient, the vulnerabilities presented in it must be taken into consideration if the level of privacy and security is high.
- CRAFT [45]: This cipher uses a 128-bit and a 64-bit block in addition to a third input known as a "tweak" for added security. It is designed primarily to be resistant to differential error attacks without neglecting an efficient resource management. Although its design contemplates resistance against attacks, potential vulnerabilities have been detected, such as weak keys and susceptibility to side-channel attacks [46]. Given that this algorithm was developed recently, further research is needed to mitigate the possible attacks that this cipher is susceptible to.
- Hummingibird-2 [47]: It is a lightweight cryptography algorithm designed for resource-constrained devices that uses a 128-bit secret key and a 64-bit initializa-

tion vector. In addition to the data encryption, this algorithm can also produce an authentication tag for each message processed. This algorithm has been susceptible to attacks capable of recovering the secret key used to encrypt messages [48].

Table 1 summarizes the features of the lightweight block ciphers algorithms discussed above.

Table 1. Features of the studied block ciphers.

Cipher Algorithm	Key Size (bits)	Block Size (bits)	Rounds
PRESENT	80, 128	64	31
SPECK	64, 72, 96, 128, 144, 1292, 256	34, 48, 64, 96, 128	22, 23, 26, 27, 28, 29, 32, 33, 34
Piccolo	80, 128	64	25, 31
CRAFT	Key: 128 Tweak: 64	64	31
Hummingbird-2	Key: 128 IV: 64	16	4

3.2. Lightweight Stream Ciphers

- ASCON [49]: This algorithm was designed specifically to be lightweight and simple to implement; it uses a 128-bit key and an initialization vector of the same length; if required, it can also produce an authentication tag of the same length. It was standardized by the National Institute of Standards and Technology (NIST) in the year 2023 after winning the CAESAR competition for its efficiency and security [50]. It was designed specifically to withstand various cryptographic attacks; however, additional measurements are required to ensure its security in practical implementations [51].
- ACORN [52]: Designed to be lightweight and efficient, this algorithm uses a 128-bit key and a 128-bit initialization vector; it can also produce an authentication tag of the same size if required. It was one of the candidates for standardization in the CAESAR competition along with the current standard, ASCON. This cipher possesses vulnerabilities to different types of cryptanalytic attacks, especially when the initialization vector is reused to encrypt messages [53]; thus, further studies are necessary to determine whether the algorithm is appropriate to ensure the security of IoT devices.
- Lizard [54]: It is a lightweight cryptographic algorithm designed for energy-constrained devices, offering a balance of security and efficiency. It uses a 120-bit key and 64-bit initialization vector. While it provides robust security against the most common cryptographic attacks, it has been shown to have vulnerabilities that can be exploited to obtain the secret key used for encryption [55].
- Fruit-80 [56]: A cryptographic algorithm designed for communications in resource-constrained environments. Its ultra-lightweight design uses an 80-bit key and a 70-bit initialization vector. It stands out as a highly efficient algorithm with major optimizations for hardware implementation. Several security analyses have detected potential vulnerabilities when it falls victim to correlation attacks [57]. Despite this, Fruit-80 offers strong resistance against known attacks and is a viable option for IoT due to its efficiency.
- TRIVIUM [58]: This algorithm is designed for the efficient operation of both hardware and software. It uses an 80-bit key and an initialization vector of the same length. Its design stands out for its simplicity, efficiency, and high speed, for which it has been the subject of security and efficient implementation research. New variants and security improvements have been proposed to mitigate cryptographic attacks that the cipher is vulnerable to [59]. These proposals highlight the efforts to reinforce stream

ciphers against potential attacks while maintaining their efficiency and, if possible, improving it.

The features of the stream ciphers mentioned above can be seen in Table 2.

Table 2. Features of the studied stream ciphers.

Cipher Algorithm	Key Size (bits)	Block Size (bits)	Rounds
ASCON	128	64, 128	128
ACORN	128	293	128
Lizard	120	121	64
Fruit-80	80	80	70
TRIVIUM	80	288	80

3.3. Security Resilience of the Selected Algorithms

Lightweight cryptography offers significant advantages over conventional options for resource-constrained devices. However, these advantages are counterbalanced by trade-offs at the cost of security. These trade-offs exist due to the time, design, and cost constraints that are present in the IoT and WSN ecosystems. Table 3 shows a comparison between the strengths and weaknesses of the selected algorithms [60–69].

Table 3. Comparison between security resilience and vulnerabilities for the selected algorithms.

Algorithm	Security Resilience	Vulnerabilities
PRESENT	Low resource consumption, strong against differential and linear cryptanalysis.	Advance cryptanalysis techniques if the number of rounds is reduced.
SPECK	Designed for efficiency, resistance to differential and linear cryptanalysis	Susceptible to certain types of differential attacks if improperly implemented.
Piccolo	Strong against differential attacks, optimized for low-resource environments.	Vulnerable to specific linear attacks and key recovery methods if the key size is not adequate.
CRAFT	Resilient against known cryptanalytic attacks	Vulnerable to brute-force attacks due to smaller key sizes in some configurations.
Hummingbird-2	High resistance to side-channel attacks, suitable for constrained devices	Potential weaknesses in key management and side-channel attack depending on implementation.
ASCON	Designed to resist side-channel attacks and provide high security. Winner of the NIST LWC competition.	Potential vulnerabilities may emerge over time.
ACORN	Strong security claims against known vulnerabilities.	Potential weaknesses in key management or implementation flaws.
Lizard	Resilient against differential and linear attacks	Potential vulnerabilities may be discovered as it is less widely studied.
Fruit-80	Resistant to several cryptanalytic attacks	Vulnerable to specific types of cryptanalyses if the implementation does not adhere to best practices.
TRIVIUM	Strong against algebraic attacks	Susceptible to algebraic attacks if key management is weak or the initialization vector is predictable.

4. Lightweight Cryptography Algorithms Analysis

In this section, the methodology used to perform the evaluation of the symmetric encryption algorithms studied is described, alongside the devices used for their implementation as well as the software and tools used. Lastly, the performance metrics used for the evaluation of the algorithms are explained. Figure 5 illustrates the methodology used for this research. Each stage of the procedure is described below.

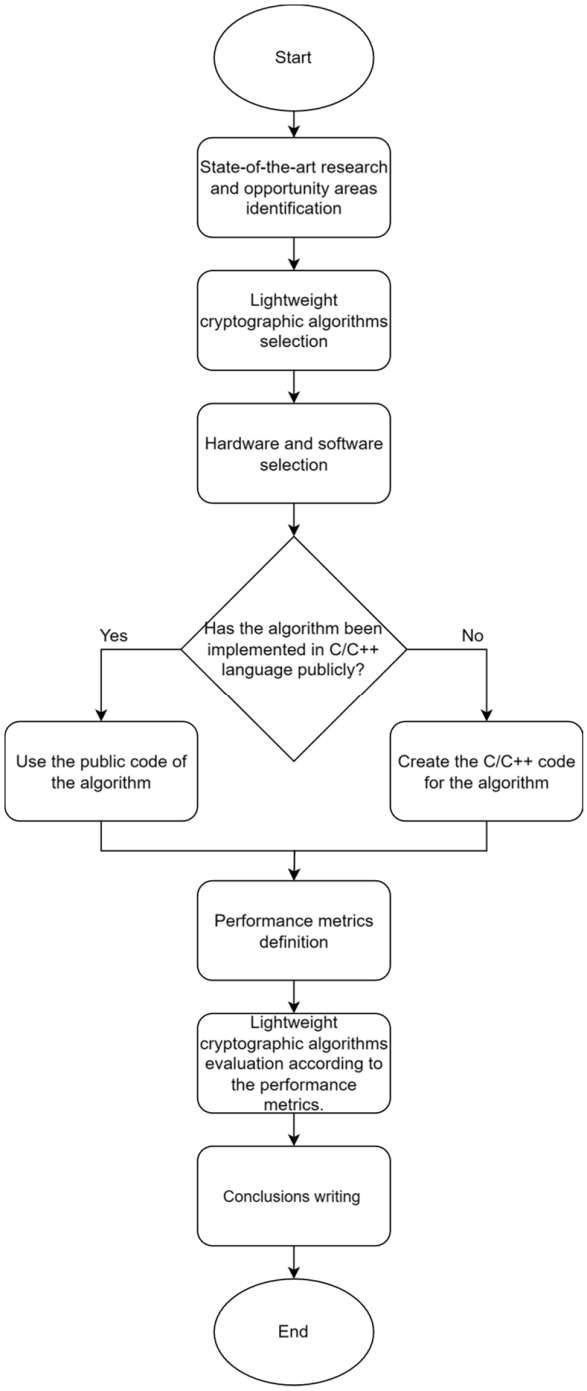


Figure 5. Flowchart of the work methodology used in this research.

4.1. State of the Art Research and Opportunity Ideas Identification

Relevant information regarding the current overview in security for embedded systems used in IoT was collected in order to have an overview about the topic. Subsequently, opportunities for improvements in the security for these systems were studied, thus defining the key challenge to be addressed: the current lack of security in embedded systems.

4.2. Lightweight Cryptographic Algorithms Selection

After reviewing the state of the art, cryptographic algorithms were selected only if they qualified as "lightweight", meaning that they have a reduced resource consumption. The selected algorithms were chosen due to being some of the most relevant and studied based on the literature reviewed [31–37]. Only 10 algorithms were selected due to time con-

siderations for this research. The selected algorithms are mentioned in Section 3 above. In addition, the international standardization and guidelines compilation, currently stated in the international standard ISO/IEC 29192 Lightweight Cryptography, describes guidelines for scaling lightweight cryptographic algorithms when used in IoT-based networks.

4.3. Hardware and Software Selection

Development boards were chosen among the most popular for IoT projects according to the state of the art methods reviewed [70–72] that had low amounts of memory and processing, but not so small that the algorithms selected could not be implemented on them. The specifications of the selected development boards are mentioned below in Table 4.

Table 4. Specifications for the development boards used.

Development Board	ESP32 Dev Module	NodeMCU ESP8266	MSP430 Launchpad	NUCLEO-G0B1RE	Arduino Nano
Data width	32-bit	32-bit	16-bit	32-bit	8-bit
Clock [MHz]	240	160	16	64	16
RAM [kB]	520	64	0.5	144	2
ROM [kB]	448				1
FLASH [kB]		512	16	512	32
Pins	30–36	30	24	64	30
Supply voltage [V]	5	5	3.6	5	7–12
I/O voltage [V]	3.3	3.3	3.6	3.6	5

4.4. Code Implementation

The code used to evaluate the lightweight cryptographic algorithms were implemented in C/C++ language, given that it is one of the most popular programming languages for IoT, along with its high efficiency. Public implementations of the algorithms were sought for reuse; if no such implementations were found, the code was then implemented based on the documentation provided by the algorithm designers.

The ASCON and SPECK algorithms were implemented using the Arduino Crypto library developed by Rhys Wheatherley [73], included in Appendix A, which provides traditional and lightweight cryptography algorithms. The PRESENT and Piccolo algorithms were implemented based on the code published on GitHub [74,75] that was modified for its use in development boards. The remaining algorithms, CRAFT, Hummingbird-2, Acorn128, Lizard, Fruit-80, and TRIVIUM, were developed based on the provided description of the algorithms according to their respective documentation.

4.5. Performance Metrics Definition

To evaluate the previously mentioned algorithms, performance metrics were defined. The definition of those metrics and how they were measured are mentioned below:

- RAM memory: The minimum amount of RAM memory required to implement an algorithm. This metric was obtained by programming each development board with a no-content code to obtain the minimum memory used by the board. After this, the same development board was programmed with each of the cryptographic algorithms, resulting in a new memory value. The difference in memory in both measurements is the desired metric. The amount of RAM memory consumed by the code is displayed on the compiler when the compilation process is finished.
- ROM/FLASH memory: The amount of ROM or FLASH memory required by the algorithm to be implemented. This parameter was obtained in the same way as the RAM memory.

- Encryption/Decryption latency: Average time required by the algorithm to produce one single byte of encrypted or decrypted data. Each of the algorithms were ran 5000 times, and the time required to carry out these executions was measured. The execution time of these iterations was calculated with a coded function in each development board, as shown in Figure 6. The time required per byte was calculated using Equation (1):

$$latency \left[\frac{\text{s}}{\text{bytes}}\right] = \frac{executionTime \text{ [s]}}{5000 * dataSize \text{ [bytes]}} \quad (1)$$

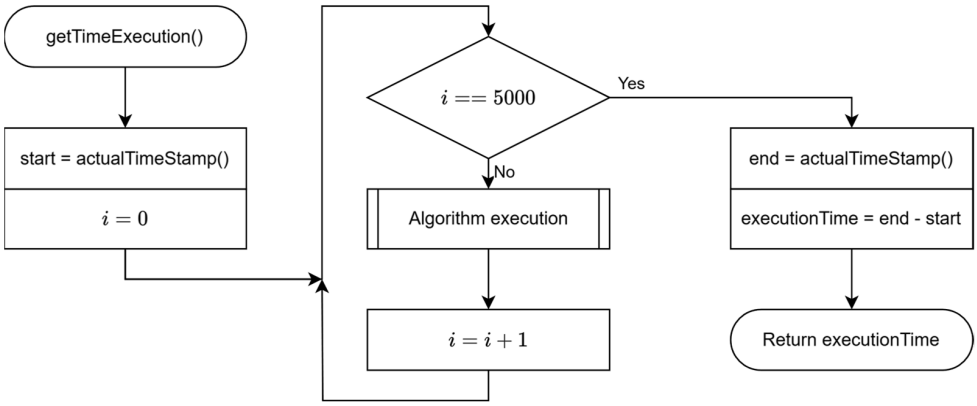


Figure 6. *executionTime* determination process.

- Encryption/Decryption throughput: The average amount of bytes that the algorithm can produce per second when running on a development board. This parameter was obtained by measuring the time required to run each of the algorithms 5000 times with the same implementation previously used to calculate the execution time for latency, as shown in Figure 6. Then, using this time and the data size of the processed message, the metric was obtained using Equation (2):

$$throughput \left[\frac{\text{bytes}}{\text{s}}\right] = \frac{5000 * dataSize \text{ [bytes]}}{executionTime \text{ [s]}} \quad (2)$$

- Current: The average value of the electrical current each development board required for encryption/decryption. Each of the algorithms were run indefinitely on each development board and powered with a 5 V voltage source with a margin of error of ± 0.01 V. The current was measured by connecting an ammeter in series between a voltage source and the development board. A diagram of this connection is shown in Figure 7.
- Power: The average transfer rate of electrical energy required by each of the algorithms to encrypt/decrypt. This parameter was calculated using the current measured mentioned above and the 5 V supply voltage value as shown in Equation (3):

$$Power \text{ [W]} = Voltage \text{ [V]} * Current \text{ [A]} \quad (3)$$

- Energy: Required energy to produce a single byte of encrypted/decrypted data. This parameter was obtained using the power measured for each algorithm and the latency that it takes to process a byte of data as shown in Equation (4):

$$Energy \left[\frac{\text{J}}{\text{byte}}\right] = Power \text{ [W]} * latency \left[\frac{\text{s}}{\text{bytes}}\right]$$

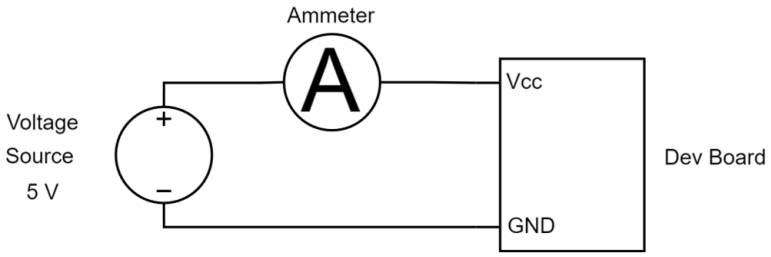


Figure 7. Electrical circuit used to measure current in the development boards.

4.6. Lightweight Cryptographic Algorithms Evaluation

For the evaluation of the selected algorithms, an ideal test scenario was used, which consisted of implementing only the code of each algorithm in the development boards previously mentioned. To verify the correct operation of the algorithms, different messages were encrypted depending on the maximum data length allowed by each block cipher, and 16-byte data was used for stream cipher testing. Once the message was encrypted, it was decrypted to obtain the original text, thus verifying their correct operation. Figure 8 illustrates the connection between the development board tested and the computer used to visualize the correct operation of the implemented algorithms. The development boards were used to execute the symmetric cryptographic algorithms and evaluation according to the performance metrics mentioned before. The development boards were connected to a computer to visualize the correct operation of the algorithms and visualize results. The implementation and visualization of the algorithms were performed using the PlatformIO extension available in the Visual Studio Code IDE, supported by the Arduino framework, for easy implementation and using its serial monitor to display results. It is worth mentioning that the same code corresponding to each algorithm was used to program each of the development boards mentioned.

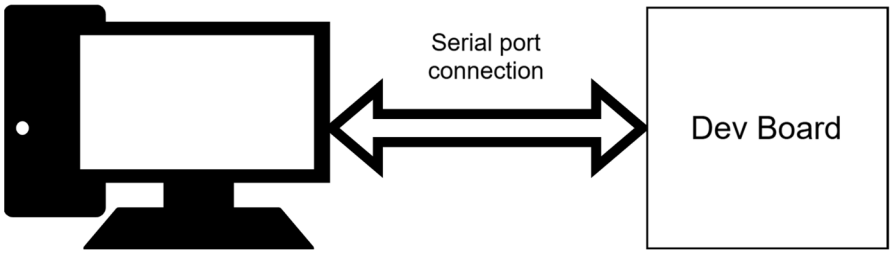


Figure 8. Communication between the development board and computer to visualize results.

4.7. Conclusions Writing

After implementing and evaluating all proposed algorithms in each of the development boards, a conclusion was drawn.

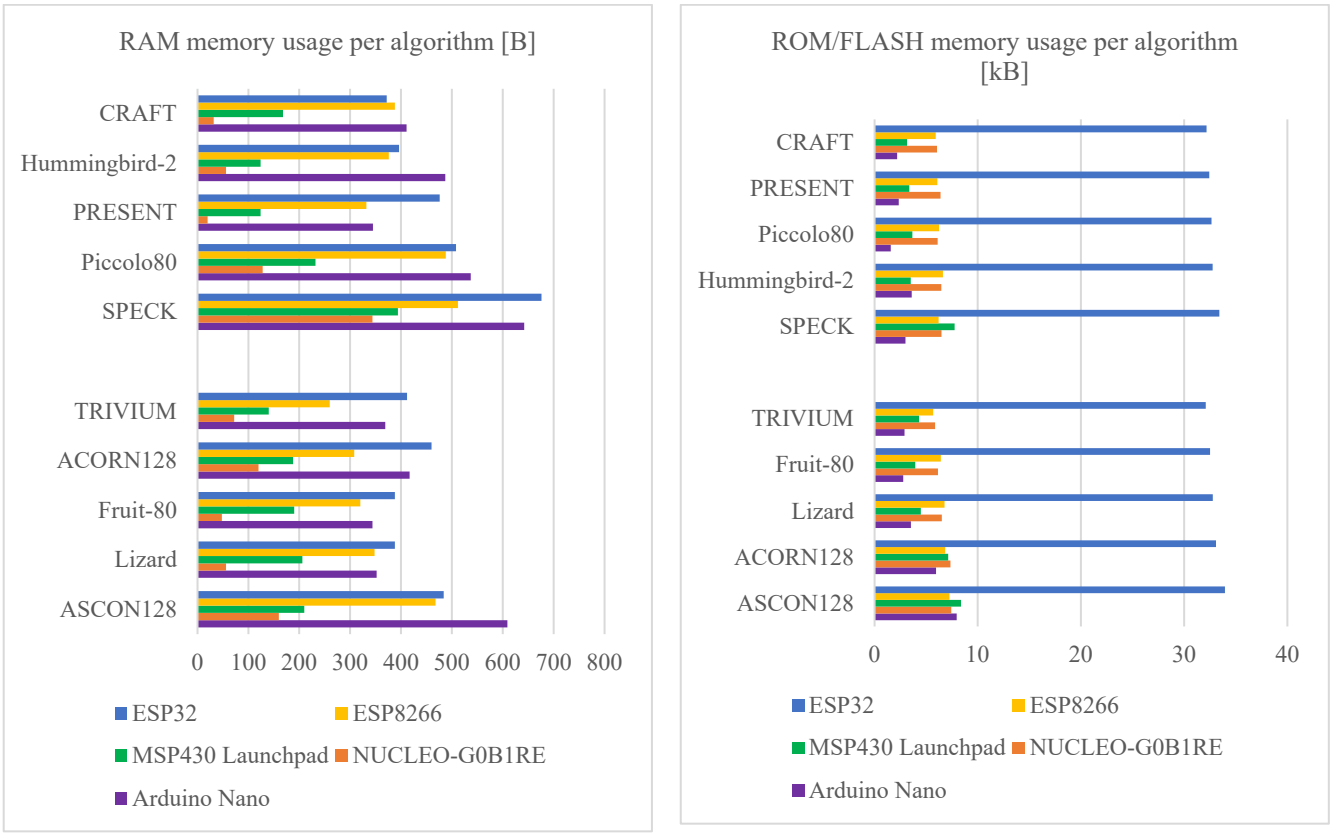
5. Results

This section shows the results obtained from the implementation of each of the symmetric encryption algorithms studied as well as their analysis, evaluation, and comparison. All the code implementations used for the evaluation of the algorithms are available in a GitHub repository link included in Appendix A. To facilitate comparison, the results are shown as graphs. In addition to this, comparison tables of all the results obtained are shown in Appendix B.

5.1. Memory Usage

Figure 9 shows the amount of RAM memory required by each symmetric algorithm, both RAM and ROM/FLASH. Focusing on Figure 9a, the results obtained reveal significant

variations among the algorithms and an even greater variation among the boards used. It can be easily noted that the development boards that require the largest amount of RAM memory to implement the algorithms are the ESP32 and Arduino Nano boards, followed closely by the ESP8266 board. On the other hand, the development boards that require the least amount of RAM memory are the NUCLEO-G0B1RE and MSP430 Launchpad boards. The code used to implement the algorithms in all of the boards is the same for each of the boards analyzed, so the difference in RAM consumption may be due to differences in the architecture of the processors. Another reason for why the consumption is different between boards is the compilation process used by the PlatformIO compiler for each of the cards; this huge difference between boards mainly occurs as the compilation performed by PlatformIO is not well optimized for the ESP family boards, especially ESP32, as it has a considerable amount of RAM. As for the comparison between algorithms, the one with the highest RAM requirement consistently across all boards is the SPECK algorithm, reaching up to 676 bytes for the ESP32 board implementation, implying that its implementation may not be ideal for applications where memory is a critical resource. On the other hand, algorithms such as PRESENT and CRAFT proved to be the most efficient on various boards. In particular, PRESENT only requires 20 bytes of RAM when implemented on the NUCLEO-G0B1RE development board, standing out as the most economical option in terms of RAM memory. Similar behavior is observed in terms of RAM requirements by the cryptographic algorithms for each of the board's implementations, so it is inferred that the memory consumption per algorithm may be strongly linked to the design used for each of the algorithms.



(**a**) (**b**)

Figure 9. Memory usage per algorithm. (**a**) RAM memory; (**b**) ROM/FLASH memory.

Similar to the previous point, Figure 9b shows the ROM/FLASH memory usage required to implement each of the cryptographic algorithms studied. The experimental results show different performance characteristics in terms of the development boards; the ESP32 platform shows a significantly higher requirement in comparison with the other boards used for all of the algorithms implemented. This can be attributed to the complexity of the architecture of the microcontroller in addition to the instruction set it uses. Conversely, the Arduino Nano board exhibits a more efficient use of ROM/FLASH memory compared with the other development boards evaluated, with implementations requiring only 1.57 KB, as is the case in the Piccolo80 algorithm. These results indicate that the Arduino Nano board is a platform that is designed to implement applications with limited memory considerations. Meanwhile, the ESP32 board is designed to support more demanding and robust applications. The remaining boards have a higher memory requirement compared with the Arduino Nano, but not as high as in the case of ESP32, making them a middle grown between the two. The latter methods are more efficient when it comes to implementing moderate algorithms.

5.2. Encryption and Decryption Latency

Figures 10–14 show the algorithm latency for processing a single byte of data for both encryption and decryption measured on the ESP32, ESP8266, MSP430 Launchpad, NUCLEO-G0B1RE, and Arduino Nano boards, respectively. Firstly, when the results among boards are compared, it can be observed that the encryption and decryption latencies are lower on the ESP32 board when compared with the other platforms evaluated. Moreover, the ESP8266 and NUCLEO-G0B1RE boards show consistently moderate times, suggesting limitations in computing power when compared with ESP32. Finally, the MSP430 Launchpad and Arduino Nano boards were the slowest platforms by a large margin when compared with the previous ones. The difference in latency times for the boards evaluated is mainly due to the difference in architecture between boards, but more importantly, the clock speed of each board evaluated; the execution times for each of the boards are related to the clock speed used, so the higher the clock speed, the smaller the latency.

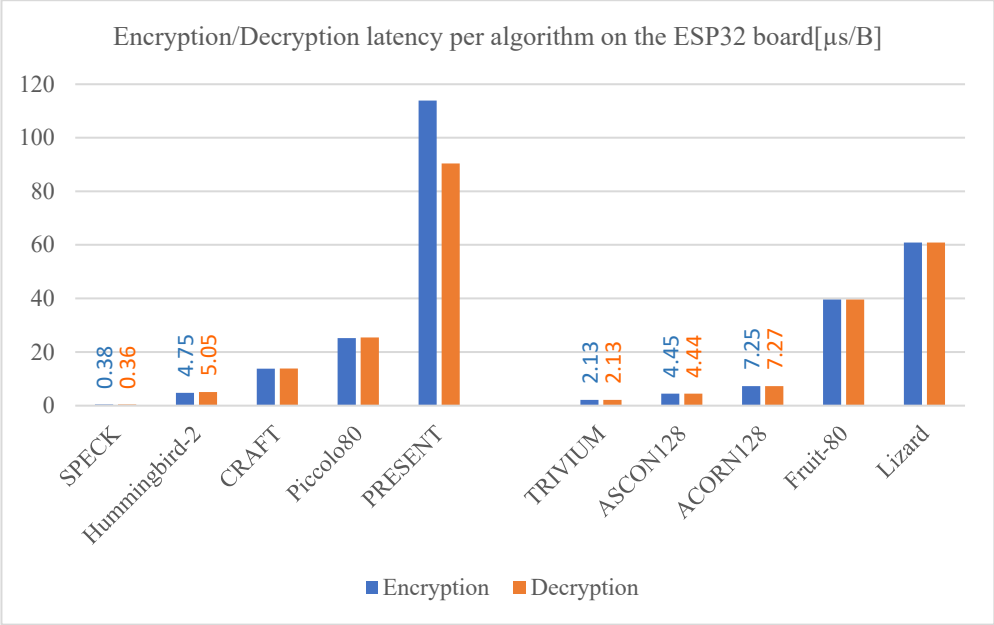


Figure 10. Encryption/Decryption latency per algorithm on the ESP32 board.

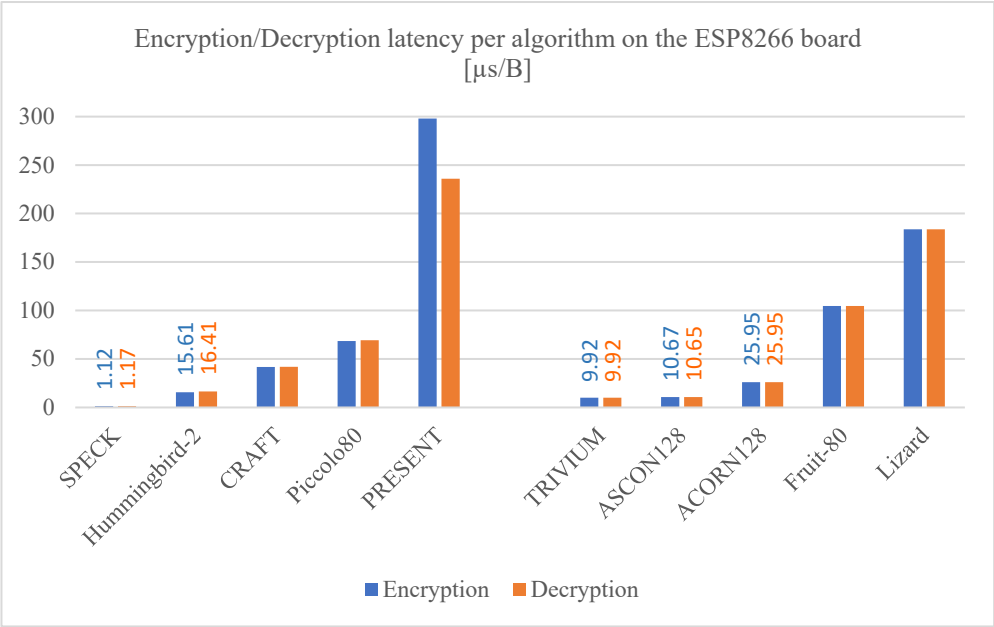


Figure 11. Encryption/Decryption latency per algorithm on the ESP8266 board.

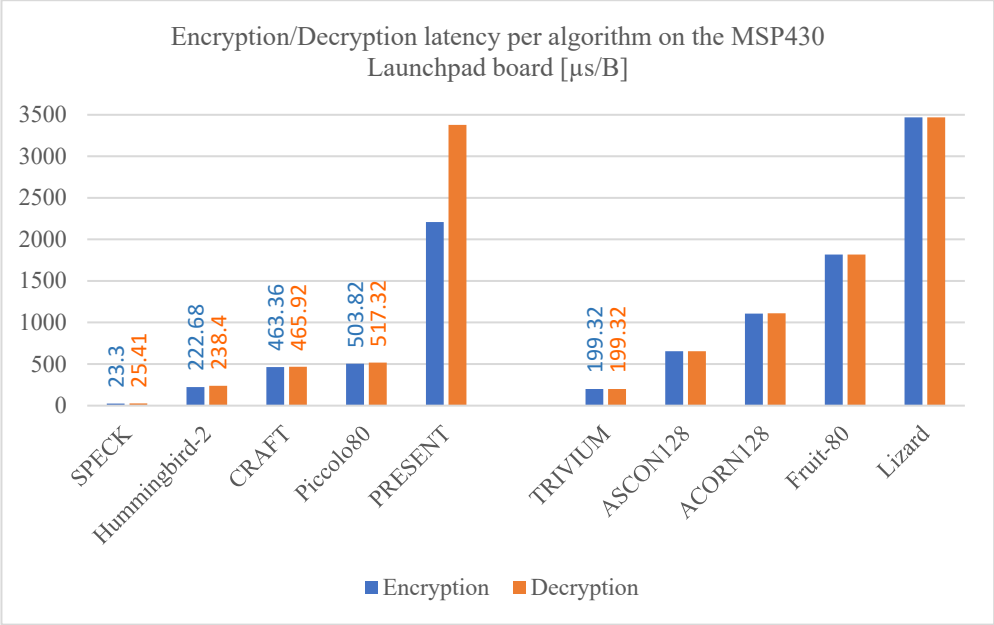


Figure 12. Encryption/Decryption latency per algorithm on the MSP430 Launchpad board.

When comparing algorithms, it is noted that SPECK is consistently the fastest of the algorithms evaluated in all platforms, which makes it an ideal algorithm for real-time applications, where fast encryption and decryption of data is critical. By contrast, algorithms such as PRESENT and lizard show significantly longer latency times, which could limit their use in time-sensitive applications, making them even more unfeasible for most applications. It is also worth noting that the execution times for encryption and decryption are similar in most cases, which is to be expected as the decryption process is inversely proportional to the encryption process.

Significant differences are noted when comparing latency among the algorithms; the PRESENT block cipher stands out as the lowest in most of the development boards evaluated, except for the NUCLEO-G0B1RE board, where the algorithm with the highest latency is lizard. This high latency means that these algorithms require more time to

process one byte of data compared with the other algorithms. On the other hand, the SPECK algorithm has the lowest latency among all algorithms evaluated in all development boards, showing great performance and suitability for low-power and real-time applications.

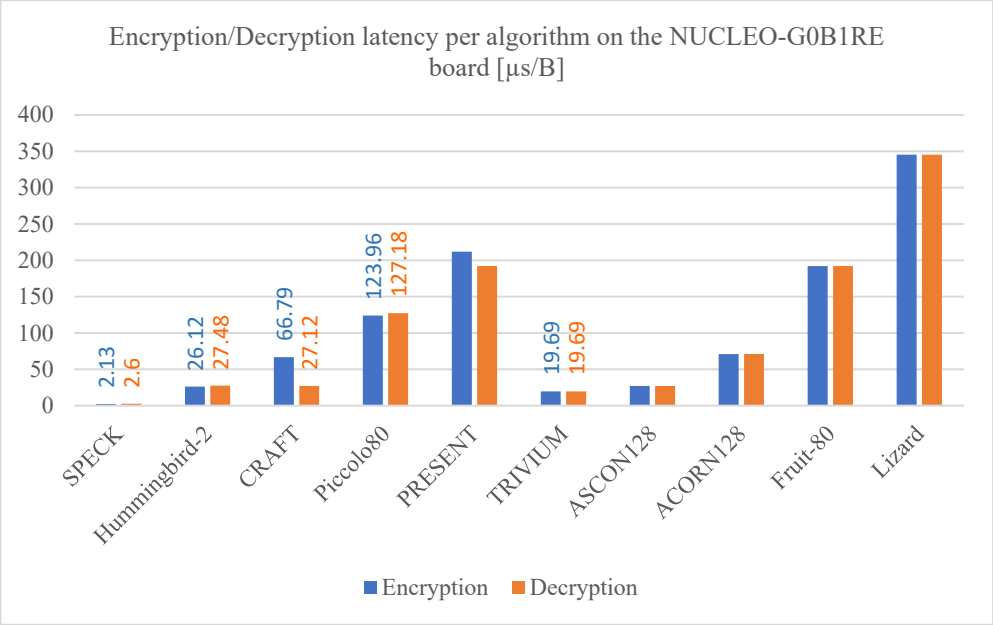


Figure 13. Encryption/Decryption latency per algorithm on the NUCLEO-G0B1RE board.

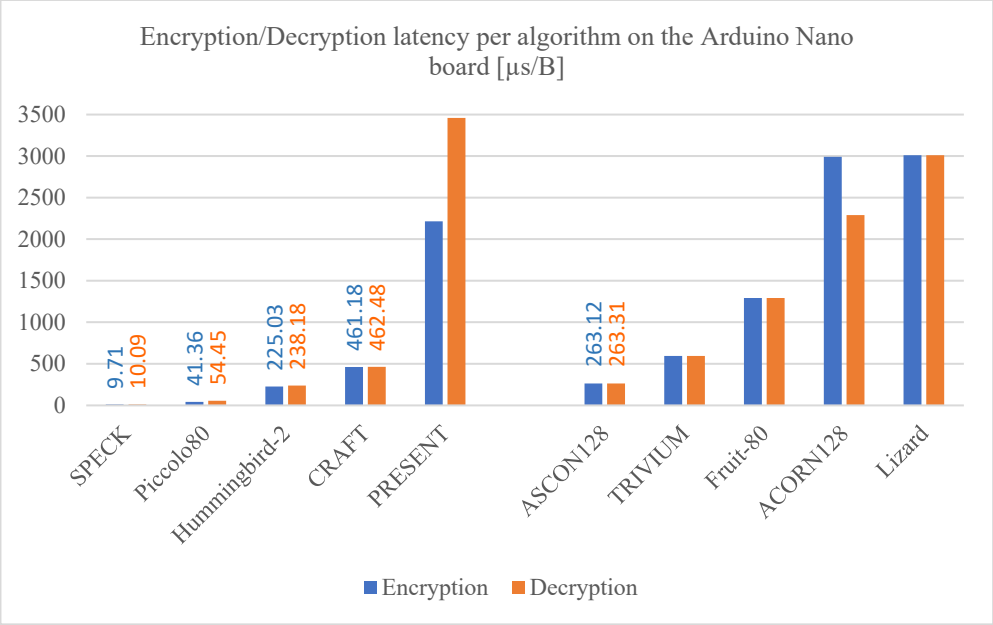


Figure 14. Encryption/Decryption latency per algorithm on the Arduino Nano board [µs/B].

5.3. Throughput for Encryption and Decryption

Figures 15–19 show the throughput for the encryption and decryption of the selected algorithms implemented on ESP32, ESP8266, MSP430 Launchpad, NUCLEO-G0B1RE, and Arduino Nano boards. Comparing these results with previous ones, latency and throughput are inversely proportional to each other, which is expected as, if less time is required to process one byte of data, more bytes can be processed in each period. When the boards' performances are compared, it is noted that ESP32 has the highest throughput capabilities among all evaluated boards. ESP8266 exhibits lower processing capabilities

than the previous one but is still adequate for most of the algorithms implemented on it. Once again, the differences in throughput capabilities across all boards are mainly attributed to the clock speed used in each board, in addition to its architecture.

The NUCLEO-G0B1RE board is placed at an intermediate level, so extra considerations regarding throughput requirements should be taken into account when implementing encryption algorithms on this platform. Finally, the MP430 Launchpad and Arduino Nano boards show significantly low throughput capabilities, which makes most of the implemented algorithms not feasible for real applications, as is the case for the PRESENT and lizard algorithms.

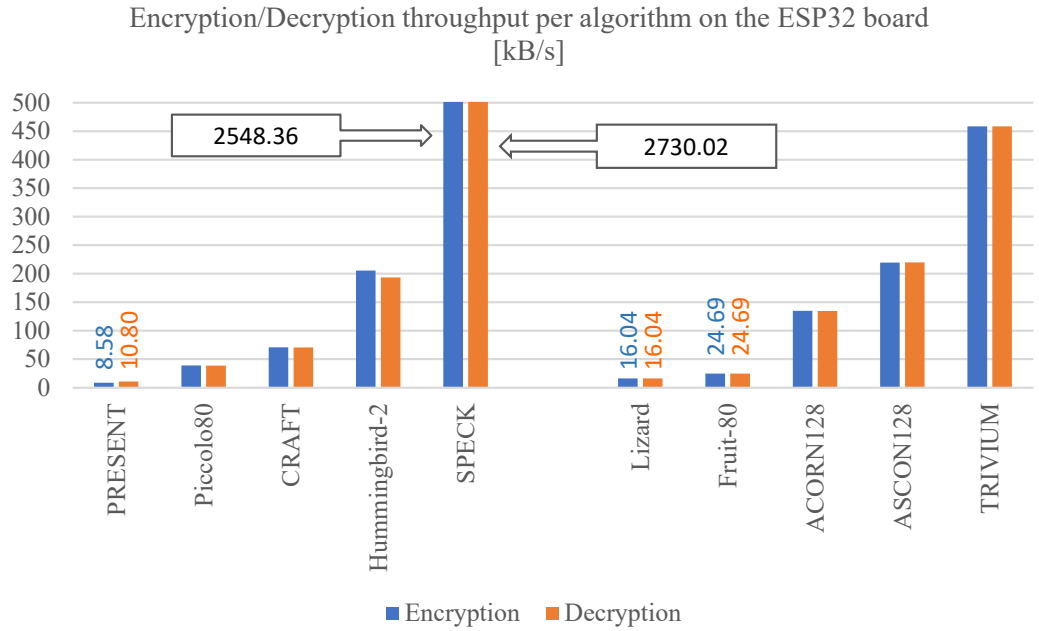


Figure 15. Encryption/Decryption throughput per algorithm on the ESP32 board.

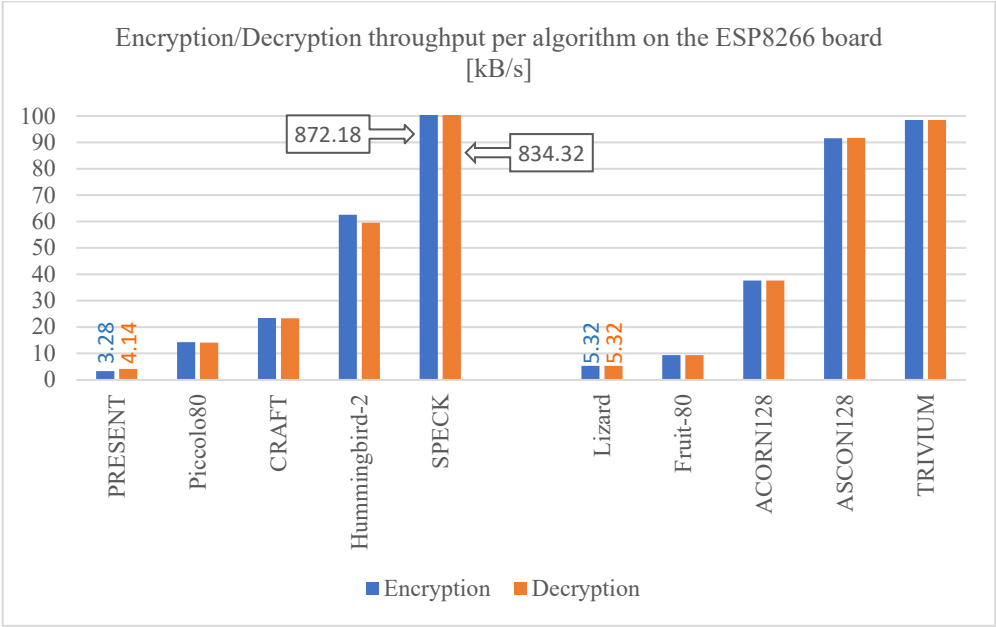


Figure 16. Encryption/Decryption throughput per algorithm on the ESP8266 board.

As for the comparison across algorithms, once again the SPECK algorithm stands out as the highest throughput capacity over all platforms with a high margin, making it a

great candidate for applications where data need to be produced in a short time. On the other hand, algorithms such as lizard and PRESENT show the worst throughput on all of the evaluated boards, especially on the MSP430 Launchpad and Arduino boards. This behavior suggests significant limitations for their implementation in applications where the production of a large volume of data in the shortest possible time is a major concern, making it even unfeasible on the beforementioned development boards due to their low throughput. The throughput capacity of each of the algorithms evaluated is different from each other, which is mainly due to their characteristic design. However, these capabilities should be taken into account for viable applications of each of the algorithms on each development board selected.

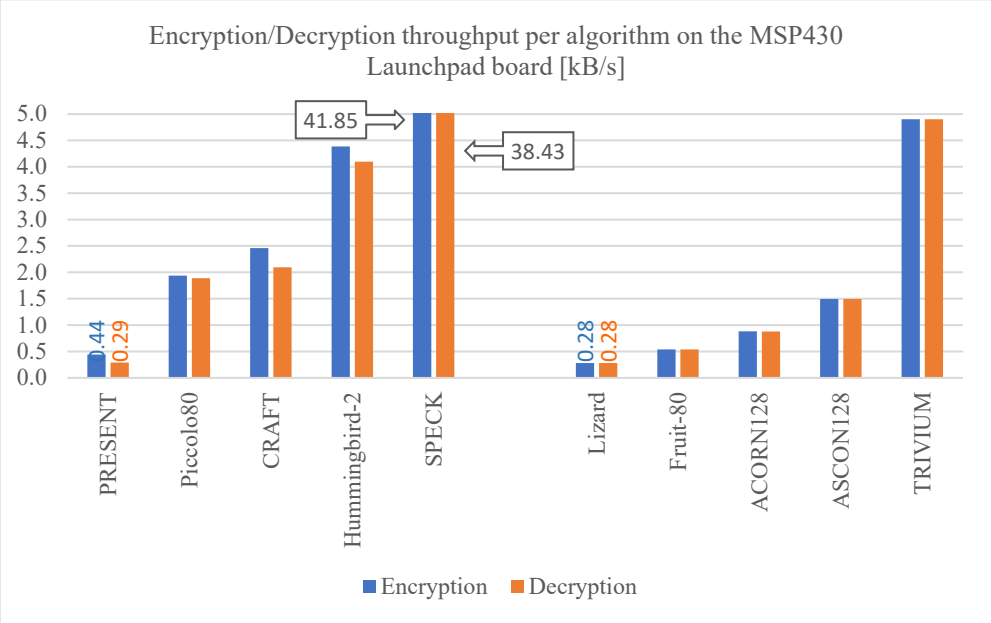


Figure 17. Encryption/Decryption throughput per algorithm on the MSP430 Launchpad board.

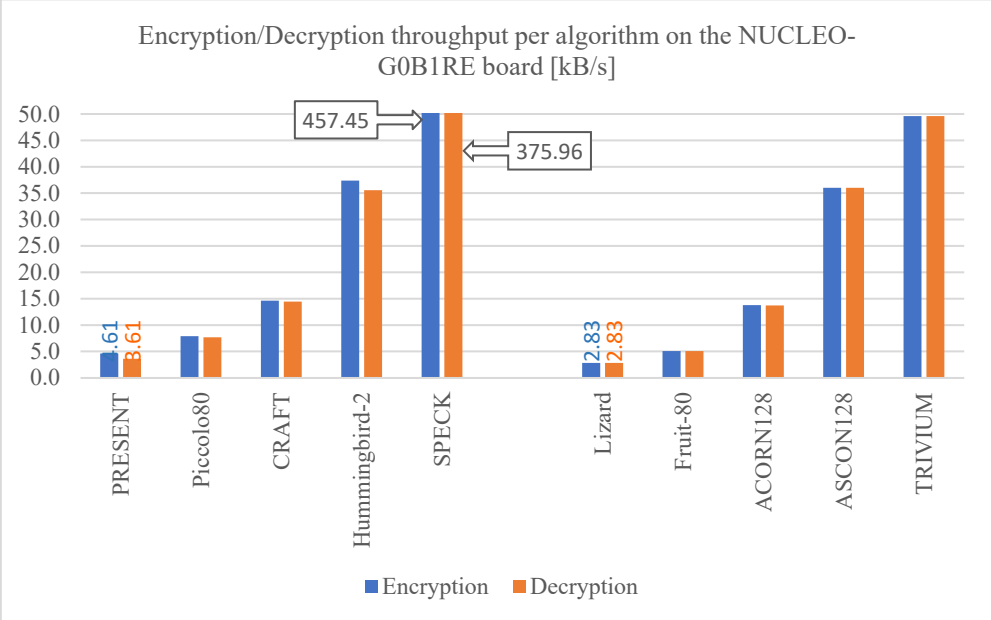


Figure 18. Encryption/Decryption throughput per algorithm on the NUCLEO-G0B1RE board.

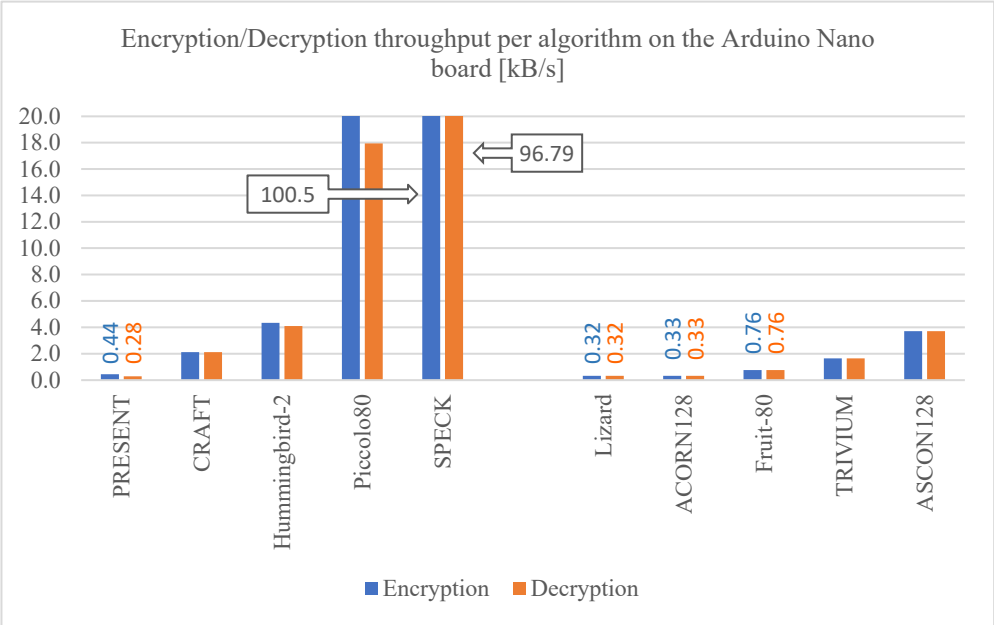
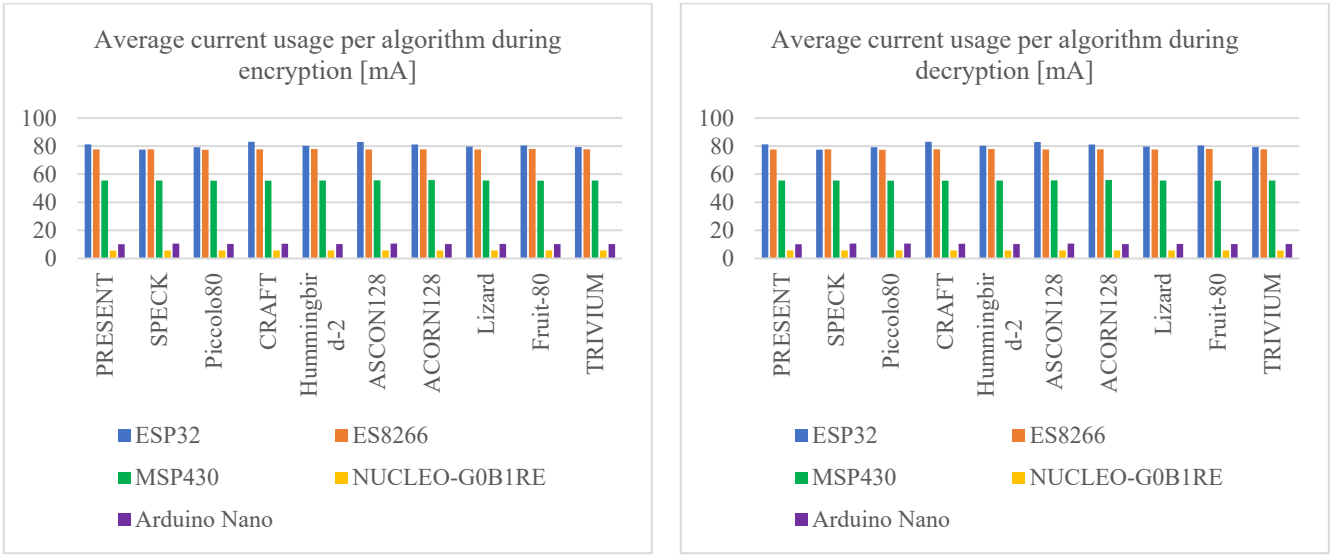


Figure 19. Encryption/Decryption throughput per algorithm on the Arduino Nano board.

5.4. Electric Current Usage

Figure 20 shows the average current usage of each development board during the execution of every algorithm for both encryption and decryption. The values obtained in each of the implementations are very similar to each other, so it is not considered that there is a significant difference in terms of the electrical current requirements among algorithms. However, if these requirements are compared between boards, clear differences are observed; the ESP32 board requires the highest electrical current consumption among all platforms studied, being closely followed by the ESP8266 board. The MSP430 Launchpad has a medium current consumption rate compared with all of the boards studied. Finally, the Arduino Nano and NUCLEO-G0B1RE boards show a significantly lower power consumption than all of the previously mentioned boards, where the latter requires on average a value of 5.577 mA to execute cryptographic algorithms.



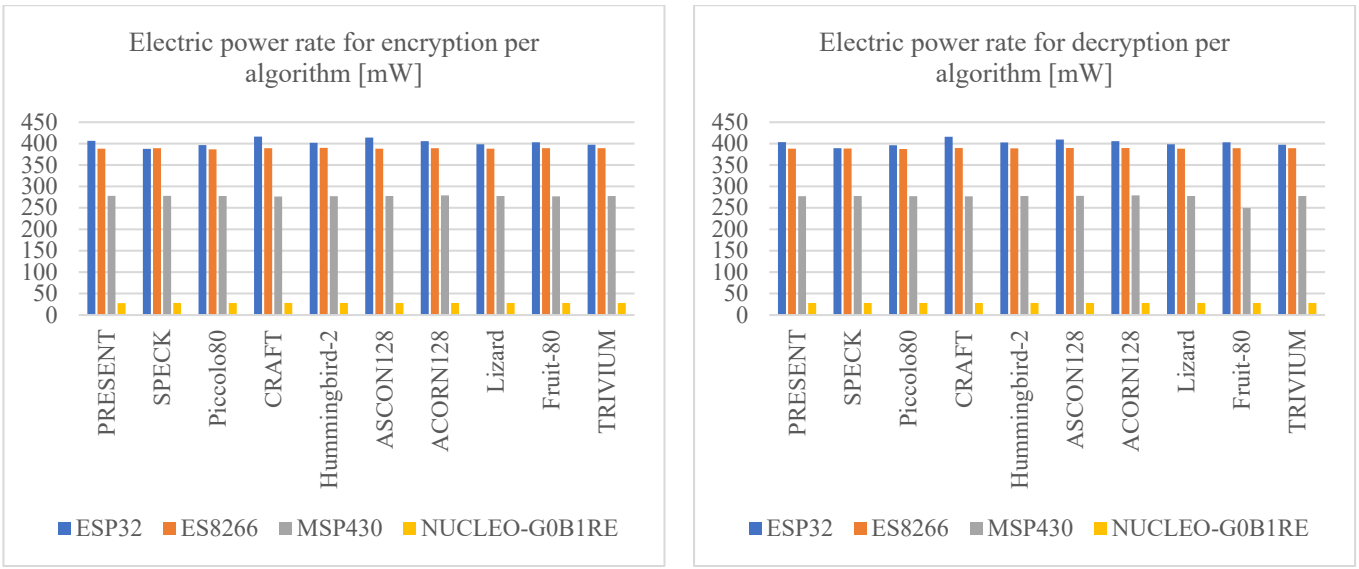
(**a**) (**b**)

Figure 20. Average current usage per algorithm. (**a**) Encryption current usage; (**b**) decryption current usage.

Similarly, the average differences between the encryption and decryption processes are remarkably small in most of the platforms, so the variation presented in the data is not considered significant. The consistency in current usage between encryption and decryption processes on most platforms indicates a well-optimized implementation of the algorithms, which is of utmost importance in energy-constrained applications.

5.5. Electric Power Rate

Figure 21 shows the power requirement for all of the development boards while running each of the algorithms for encryption and decryption. As in Figure 20, the power required to run any of the algorithms is similar across all of them, so there is no significant difference in power required per algorithm. The difference in power required is noted when the development boards are compared; the NUCLEO-G0B1RE and Arduino Nano boards are the ones that require less electric power compared with the other boards studied, maintaining a remarkably low and consistent power consumption for both encryption and decryption so that they are better options when power restrictions are present. The ESP32 and ESP8266 boards have a higher power requirement compared with the other boards. The ESP family boards do not show significant differences in terms of power; however, the ESP32 board requires slightly more compared with the ESP8266 board. It is particularly noted that the NUCLEO-G0B1RE maintains a more efficient and stable power consumption compared with the other boards evaluated, which positions it as an optimal choice when power efficiency is critical.



(**a**) (**b**)

Figure 21. Electric power rate per algorithm. (**a**) Encryption power rate; (**b**) decryption power rate.

5.6. Energy Usage per Byte

Figures 22–26 provide the required energy to process a single byte of data, both encryption and decryption, for each cryptographic algorithm on the development boards evaluated. The results reveal significant differences in energy required among the algorithms, as well as among the development boards used.

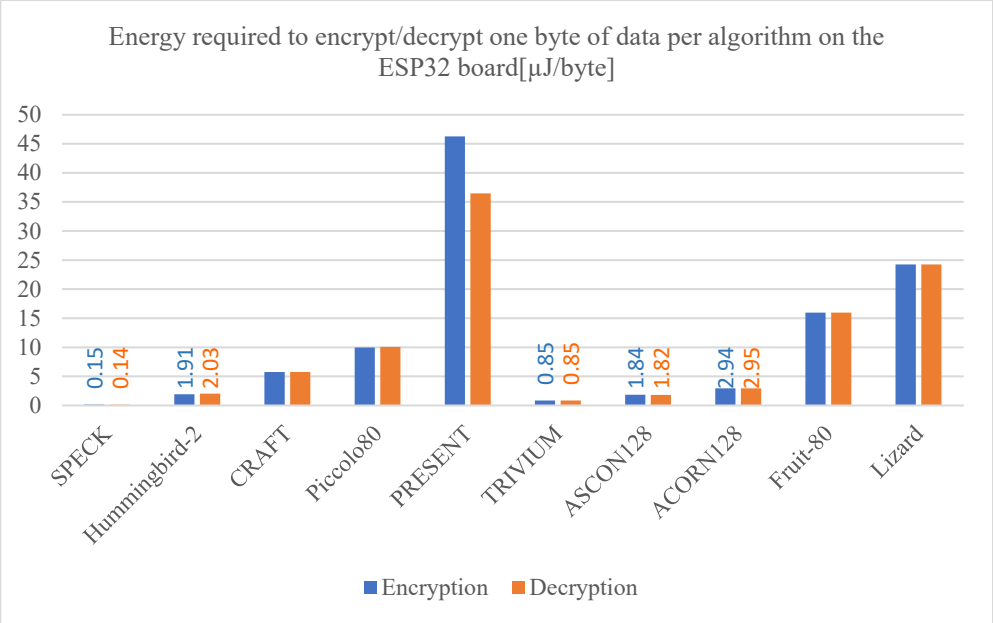


Figure 22. Energy required to encrypt/decrypt one byte of data per algorithm on the ESP32 board.

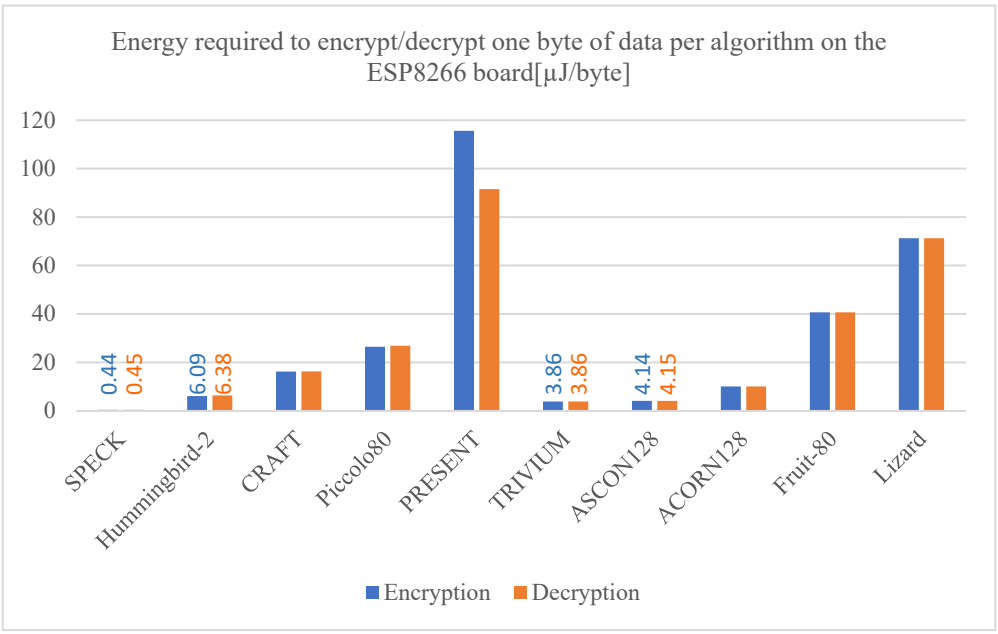


Figure 23. Energy required to encrypt/decrypt one byte of data per algorithm on the ESP8266 board.

Comparing the results across boards, it is observed that the NUCLEO-G0B1RE board presents the lowest energy requirement compared with the other evaluated boards, followed by ESP32. MSP430 Launchpad presents a significantly higher energy requirement compared with all boards, which is a crucial factor to consider in the design of energy-constrained IoT systems. This lower efficiency is directly related to the encryption and decryption latencies that each of the algorithms and boards have, in addition to the power requirements for encryption and decryption. This leads to higher energy consumption. Likewise, the ESP32 and ESP8266 boards have lower energy consumption due to them, despite requiring more electrical power to perform their operations. Comparing the results across all of the algorithms implemented on the development boards, it is observed that SPECK proves once again to be one of the most energy-efficient algorithms on all evaluated platforms, requiring only 0.06 µJ on the NUCLEO-G0B1RE board, also showing similarly

efficient values on the other platforms. In contrast, PRESENT exhibits one of the highest energy consumptions on all platforms, with particularly notable values on MSP430 Launchpad. The differences in energy requirements are mainly due to the execution time of each algorithm since, as discussed before, this is strongly linked to the energy power consumption of each of the algorithms. It is worth noting that the PRESENT algorithm presented a significant disparity in terms of encryption energy requirements compared with decryption, being almost 50% on the MSP430 Launchpad board, something that had not been presented in the previous figures in such a drastic way.

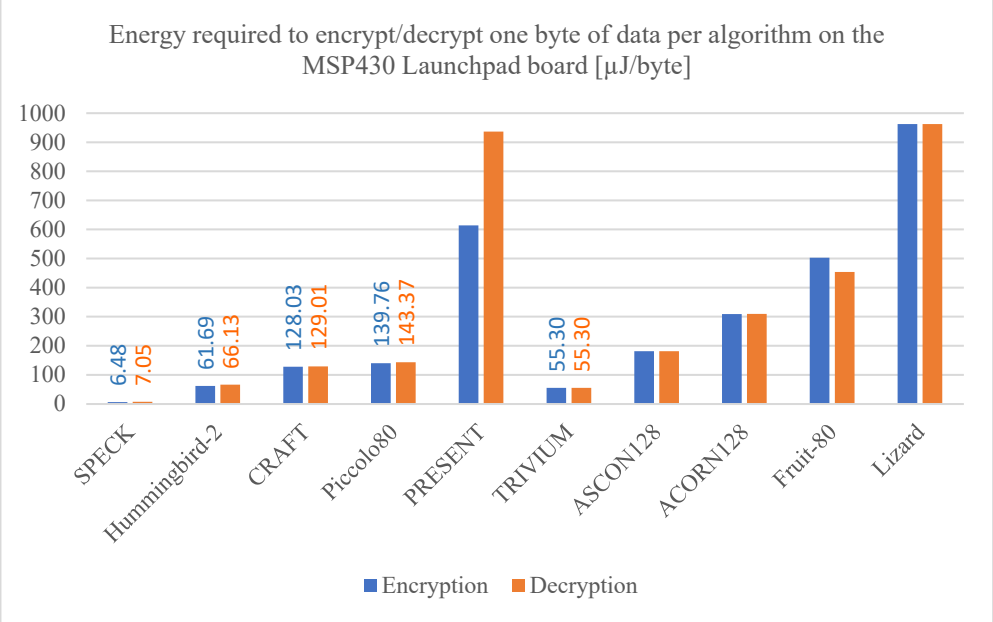


Figure 24. Energy required to encrypt/decrypt one byte of data per algorithm on the MSP430 Launchpad board.

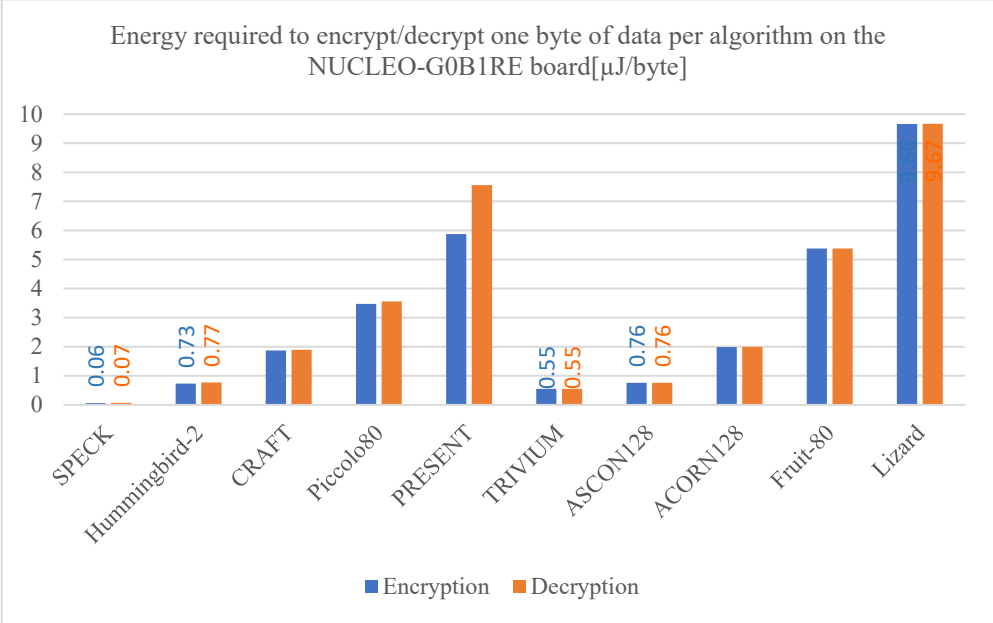


Figure 25. Energy required to encrypt/decrypt one byte of data per algorithm on the NUCLEO-G0B1RE board.

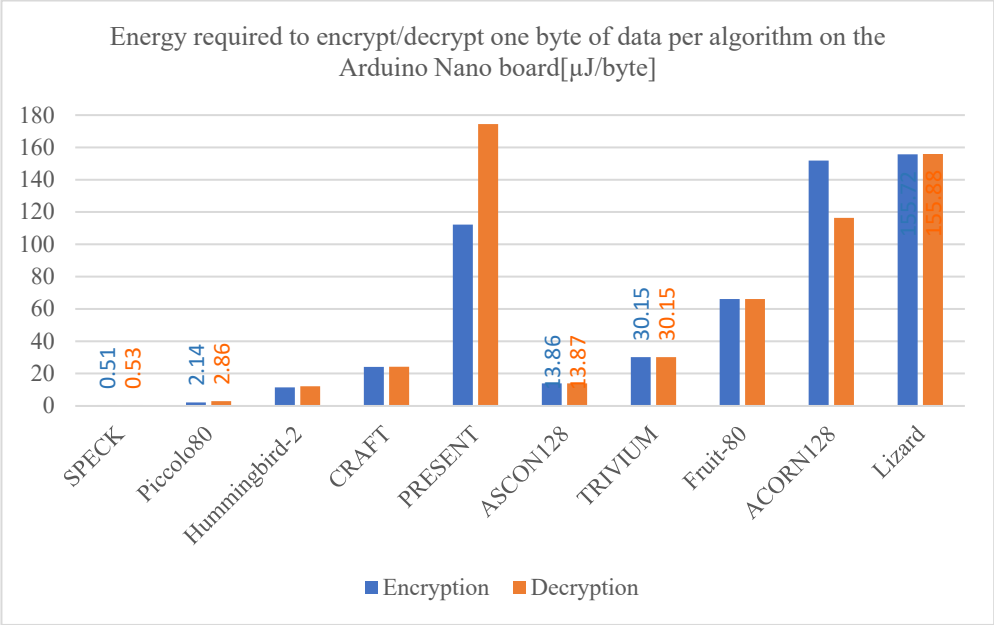


Figure 26. Energy required to encrypt/decrypt one byte of data per algorithm on the Arduino Nano board [µJ/B].

6. Discussion

6.1. Performance Discussion of Lightweight Symmetric Algorithms

From the results, it can be inferred that the SPECK and TRIVIUM symmetric algorithms exhibit remarkable memory efficiency while maintaining high processing speed, even surpassing the recent standard, ASCON. In contrast, PRESENT, lizard, and Fruit-80 face challenges in terms of efficiency and processing power. Additionally, although SPECK is energy-efficient and fast, it requires higher RAM and ROM/FLASH memory consumption for implementation. This makes SPECK a suitable choice for implementations that prioritize energy efficiency, regardless of memory constraints. On the contrary, algorithms such as PRESENT exhibit exceptionally low RAM/FLASH memory requirements. However, this advantage is counterbalanced by a high energy consumption and high latency. Consequently, algorithm selection depends on the specific IoT application and the device's memory constraints. For instance, the ASCON standard represents a moderate-consumption algorithm that strikes a balance between resource consumption and higher computational requirements.

To facilitate the decision making of developers, Figure 27 shows a comparison matrix of the resource consumption versus throughput. This matrix illustrates the advantages and disadvantages of the cryptographic algorithms evaluated and complements the results obtained. This classification helps in the selection of the most suited algorithm according to the specific requirements of each IoT application, highlighting how memory energy consumption and speed can influence this decision.

6.2. Symmetric and Asymmetric Encryption Integration

While symmetric encryption offers faster computing speed and better performance to encrypt large amounts of data, it generally provides a lower level of security compared with asymmetric encryption. A promising alternative to take advantage of both techniques is the integration of both types of cryptography in a single system; this solution is called a hybrid cryptosystem. This approach combines the efficiency and speed of symmetric encryption for encrypting large volumes of data while enhancing the system's security through the definition of a secret key using a key exchange method via asymmetric encryption [76].

However, this strategy increases computational complexity, which can be challenging for resource-constrained IoT devices [77,78].

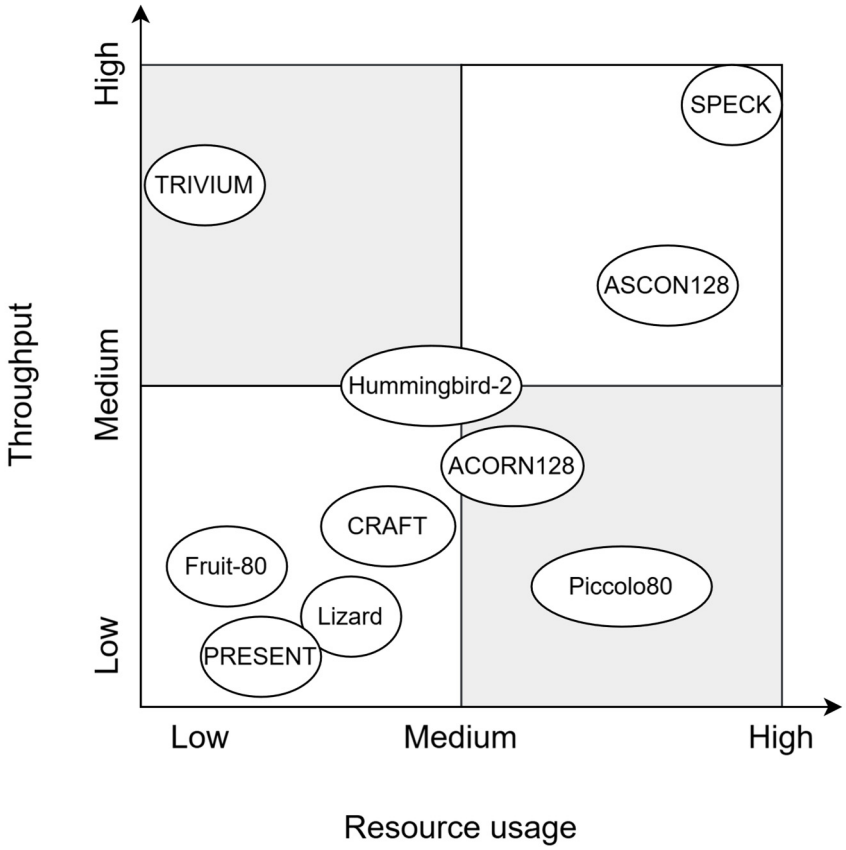


Figure 27. Resource usage vs throughput matrix.

Another innovative approach is the implementation of identity-based encryption (IBE) and identity-based broadcast encryption (IBBE) [79,80]. These methods eliminate the need for complex public key infrastructures, making them especially attractive for IoT devices with computational and storage constraints. These techniques can improve security while reducing overhead, aligning with the lightweight requirements demanded by IoT systems.

In addition to exploring hybrid and identity-based methods, future studies should include specific security assessments for lightweight symmetric algorithms, as such assessments are critical for ensuring the long-term reliability and security of lightweight cryptographic solutions in IoT applications.

7. Conclusions

Lightweight cryptography is an indispensable security mechanism for a secure implementation of Internet of Things (IoT), given that it is of major importance to guarantee the security and privacy for the data transmitted between devices across these networks. A total of ten lightweight symmetric algorithms, half block ciphers and half stream ciphers, were evaluated, focusing mainly on their use in development boards typically employed in IoT environments. The evaluation of these algorithms was successful, as all the chosen algorithms were correctly implemented and all performance metrics defined were measured on the selected development boards. As a by-product of this research, a repository was developed that contains implementations of the mentioned algorithms in C++ language, which are implemented as C++ libraries so that these can be adapted to any platform that uses this programming language, as is the case of the development boards compatible with the Arduino platform. The results of this evaluation have provided

valuable information on the performance and characteristics of each algorithm, highlighting the strengths as well as weaknesses of each algorithm with respect to its resource requirements. The memory usage evaluation shows significant differences in RAM and ROM/FLASH requirements among boards and algorithms. Boards such as the ESP32 and Arduino Nano boards exhibit higher RAM consumption, which is attributed to the microcontroller architecture and less optimized compiler configurations. On the other hand, the NUCLEO-G0B1RE and MSP430 Launchpad boards require minimal RAM, making them better suited for memory-constrained environments. Similarly, algorithms such as SPECK have more memory requirements, while PRESENT is efficient in memory usage, especially in resource-constrained boards. In terms of encryption and decryption latency, ESP32 consistently stands out against other boards due to its higher clock speed and complex architecture, making it an ideal choice for real-time applications.

Ciphers such as SPECK and TRIVIUM prove to be very efficient in memory usage while remaining fast, even above the recent standard, ASCON. In contrast, PRESENT, lizard, and Fruit-80 have challenges in terms of efficiency and processing power. It was observed that algorithms such as SPECK stand as being energy-efficient and having fast processing; however, this is traded off by a higher RAM and ROM/FLASH memory consumption required to be implemented, making this algorithm viable for implementations that require energy efficiency, regardless of memory consumption. On the contrary, algorithms such as PRESENT show an exceptionally low RAM/FLASH memory requirement, but this is counterbalanced by a high energy consumption and high latency. For this reason, this algorithm is only viable for applications where devices have memory restrictions. There are also moderate consumption algorithms, such as the ASCON standard, with a balance between resource consumption and the level of security they offer. Due to its high-complexity design, it offers greater security at the cost of higher requirements; for this reason, this kind of algorithm is better suited for IoT devices with energy constraints. Lastly, algorithms with high resource requirements and high latency, such as PRESENT, are not optimal for applications with strict resource constraints; instead, these are better suited for applications where confidentiality and privacy of the data transmitted are critical.

The energy requirement analysis highlights the trade-offs between power efficiency and computational performance. While ESP32 and ESP8266 consume more energy, their lower consumption per byte is due to shorter processing times. In contrast, NUCLEO-G0B1RE shows exceptional energy efficiency, positioning it as the most suitable choice for power-constrained systems.

The performance of the development boards evaluated also have potential and practical applications; ESP32′s combination of high throughput and low latency makes it suitable for real-time applications that require fast encryption and decryption, such as IoT gateways and smart home systems. Similarly, the ESP8266 board with moderate latencies and throughput is ideal for IoT applications that need secure communication but operate in less demanding environments, such as wearable devices and smart sensors. The NUCLEO-G0B1RE board excels in power efficiency, making it ideal for power-constrained systems such as remote environmental sensors and battery-powered devices. The Arduino Nano board, with its balanced performance, is suitable for scenarios where power conservation is critical, even at the cost of latency, like embedded medical devices and wireless sensor networks. Lastly, the MSP430 Launchpad constrained-resources configuration is ideal for scenarios where resource management is critical, like low-cost prototyping and testing.

It should be noted that, while the results shown provide valuable guidance for the selection of cryptographic algorithms, there is still a need for further research to address the identified limitations and vulnerabilities. In addition, a comprehensive approach is required that not only takes into account cryptographic security but other challenges that

arise in IoT, such as authentication, key management, protection against network attacks, and denial of service, among others.

As future research, there is a need for the evaluation of more cryptographic algorithms under development using more devices with different specifications, as well as the consideration of asymmetric encryption algorithms as another alternative to protect transmitted data, considering that, due to the rapid evolution of cyber threats, it is essential to keep up with research and development in the field of lightweight cryptography. The research in the field of asymmetric cryptography is low in comparison with symmetric cryptography due to the higher complexity and extremely challenging adaptability innovations for lightweight options of the first mentioned algorithms. Also, there are more security mechanisms, such as authentication and intrusion detection, that are needed for the secure implementation of an IoT system; for this reason, is necessary to adopt these algorithms in conjunction with cryptography to ensure a higher level of security in IoT systems.

Lastly, while these mechanisms do help in protecting IoT systems against threats, it is crucial that security education and awareness is promoted among developers, users, and stakeholders in the IoT ecosystem, as all of these mechanisms are useless if people are not aware of the risks that exist and how to identify them in order to avoid compromised security incidents.

Author Contributions: Conceptualization, J.S.-C., E.R.-I. and J.M.-S.; methodology, E.R.-I. and A.C.-A.; software, J.S-C. and A.E.-R.; validation, J.S.-C., E.R.-I. and J.V.-C.; formal analysis, J.S.-C., E.R.-I. and J.V.-C.; investigation, J.S.-C. and J.M.-S.; resources, E.R.-I.; data curation, J.S.-C. and J.M.-S.; writing-original draft preparation, J.S.-C.; writing-review and editing, E.R.-I., J.V.-C. and A.E.-R.; visualization, A.E.-R., A.C.-A. and J.M.-S.; supervision, E.R.-I. and A.E.-R.; project administration, E.R.-I., funding acquisition, E.R.-I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: No applicable.

Informed Consent Statement: No applicable.

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed at the corresponding author.

Acknowledgments: Special thanks to the Consejo Nacional de Ciencias y Tecnologías (CONACYT) for their support with the national scholarship assigned to CVU 1232870.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

The GitHub repository that includes the code and libraries for implementing the algorithms on the studied device boards: https://github.com/AleczSC/LightweightCryptography (accessed on 30 October 2024).

The Crypto library used to evaluate the ASCON and SPECK algorithms: https://rweather.github.io/arduinolibs/crypto.html (accessed on 19 April 2024).

Appendix B

Algorithm performance comparison tables.

Table A1. Implementation results of the cryptographic algorithms on the ESP32 board.

Algorithm	RAM Memory [B]	ROM/FLASH Memory [kB]	Encryption Latency [µs/B]	Encryption Throughput [kB/s]	Encryption Electric Current [mA]	Encryption Electric Power [mW]	Encryption Energy [µJ/B]	Decryption Latency [µs/B]	Decryption Throughput [kBµs]	Decryption Electric Current [mA]	Decryption Electric Power [mW]	Decryption Energy [µJ/B]
PRESENT	476	32.441	113.85	8.58	81.2	406.49	46.28	90.39	10.80	80.7	403.34	36.46
SPECK	676	33.418	0.38	2548.36	77.5	387.89	0.15	0.36	2730.02	77.8	389.16	0.14
Piccolo80	508	32.652	25.15	38.82	79.2	396.40	9.97	25.41	38.43	79.1	396.13	10.07
CRAFT	372	332.176	13.78	70.87	83.1	416.25	5.74	13.85	70.52	83.2	415.75	5.76
Hummingbird-2	396	32.758	4.75	205.52	80.2	401.72	1.91	5.05	193.38	80.5	402.58	2.03
ASCON128	484	33.961	4.45	219.43	82.9	414.00	1.84	4.44	219.81	82	409.51	1.82
ACORN128	460	33.082	7.25	134.65	81.1	405.74	2.94	7.27	134.41	81.1	405.74	2.95
Lizard	388	32.781	60.87	16.04	79.7	398.26	24.24	60.87	16.04	79.7	398.26	24.24
Fruit-80	388	32.520	39.56	24.69	80.5	403.14	15.95	39.56	24.69	80.5	403.14	15.95
TRIVIUM	412	32.098	2.13	458.27	79.3	397.29	0.85	2.13	458.27	79.3	397.29	0.85

Table A2. Implementation results of the cryptographic algorithms on the ESP8266 board.

Algorithm	RAM Memory [B]	ROM/FLASH Memory [kB]	Encryption Latency [µs/B]	Encryption Throughput [kB/s]	Encryption Electric Current [mA]	Encryption Electric Power [mW]	Encryption Energy [µJ/B]	Decryption Latency [µs/B]	Decryption Throughput [kBµs]	Decryption Electric Current [mA]	Decryption Electric Power [mW]	Decryption Energy [µJ/B]
PRESENT	332	6.09	297.98	3.28	77.6	388.00	115.62	235.83	4.14	77.5	388.12	91.53
SPECK	512	6.21	1.12	872.18	77.7	389.12	0.44	1.17	834.32	77.7	388.58	0.45
Piccolo80	488	6.26	68.47	14.26	77.4	386.69	26.48	69.31	14.09	77.4	387.39	26.85
CRAFT	388	5.93	41.73	23.40	77.7	389.20	16.24	41.89	23.31	77.8	389.70	16.32
Hummingbird-2	376	6.62	15.61	62.56	77.9	389.97	6.09	16.41	59.50	77.7	388.97	6.38
ASCON128	468	7.26	10.67	91.56	77.6	388.23	4.14	10.65	91.67	77.9	389.58	4.15
ACORN128	308	6.84	25.95	37.63	77.7	389.20	10.10	25.95	37.64	77.9	389.66	10.11
Lizard	348	6.78	183.69	5.32	77.6	388.00	71.27	183.69	5.32	77.6	388.00	71.27
Fruit-80	320	6.43	104.52	9.34	77.9	389.34	40.69	104.52	9.34	77.9	389.34	40.69
TRIVIUM	260	5.68	9.92	98.49	77.7	389.28	3.86	9.92	98.48	77.7	389.28	3.86

Table A3. Implementation results of the cryptographic algorithms on the MSP430 Launchpad board.

Algorithm	RAM Memory [B]	ROM/FLASH Memory [kB]	Encryption Latency [µs/B]	Encryption Throughput [kB/s]	Encryption Electric Current [mA]	Encryption Electric Power [mW]	Encryption Energy [µJ/B]	Decryption Latency [µs/B]	Decryption Throughput [kBµs]	Decryption Electric Current [mA]	Decryption Electric Power [mW]	Decryption Energy [µJ/B]
PRESENT	124	3.379	2209.15	0.44	55.52	278.04	614.24	3378.05	0.29	55.44	277.31	936.77
SPECK	394	7.768	23.3	41.85	55.5	277.94	6.48	25.41	38.43	55.4	277.44	7.05
Piccolo80	232	3.668	503.82	1.94	55.38	277.40	139.76	517.32	1.89	55.36	277.13	143.37
CRAFT	168	3.148	463.36	2.46	55.34	276.31	128.03	465.92	2.10	55.41	276.88	129.01
Hummingbird-2	124	3.514	222.68	4.39	55.43	277.04	61.69	238.4	4.10	55.49	277.39	66.13
ASCON128	210	8.377	653.26	1.49	55.54	277.76	181.45	653.45	1.49	55.56	277.80	181.53
ACORN128	188	7.141	1107.06	0.88	55.78	278.96	308.82	1110.2	0.88	55.76	278.91	309.65
Lizard	206	4.484	3468.17	0.28	55.49	277.56	962.63	3468.17	0.28	55.49	277.56	962.63
Fruit-80	190	3.951	1818.05	0.54	55.35	276.64	502.94	1818.05	0.54	49.97	249.75	454.06
TRIVIUM	140	4.313	199.32	4.90	55.47	277.46	55.30	199.32	4.90	55.47	277.46	55.30

Table A4. Implementation results of the cryptographic algorithms on the NUCLEO-G0B1RE board.

Algorithm	RAM Memory [B]	ROM/FLASH Memory [kB]	Encryption Latency [µs/B]	Encryption Throughput [kB/s]	Encryption Electric Current [mA]	Encryption Electric Power [mW]	Encryption Energy [µJ/B]	Decryption Latency [µs/B]	Decryption Throughput [kBµs]	Decryption Electric Current [mA]	Decryption Electric Power [mW]	Decryption Energy [µJ/B]
PRESENT	20	6.395	211.94	4.61	5.56	27.72	5.88	270.5	3.61	5.58	27.94	7.56
SPECK	344	6.496	2.13	457.45	5.59	28.00	0.06	2.6	375.96	5.59	27.99	0.07
Piccolo80	128	6.109	123.96	7.88	5.59	28.00	3.47	127.18	7.68	5.58	27.95	3.55
CRAFT	32	6.047	66.79	14.62	5.59	28.01	1.87	67.57	14.45	5.59	28.00	1.89
Hummingbird-2	56	6.477	26.12	37.38	5.59	28.00	0.73	27.48	35.54	5.59	28.00	0.77
ASCON128	160	7.430	27.12	36.00	5.59	28.00	0.76	27.12	36.01	5.59	28.00	0.76
ACORN128	120	7.352	70.92	13.77	5.59	28.00	1.99	71.15	13.72	5.59	28.00	1.99
Lizard	56	6.523	345.22	2.83	5.59	27.99	9.66	345.23	2.83	5.59	28.00	9.67
Fruit-80	48	6.141	192.02	5.09	5.59	28.00	5.38	192.02	5.09	5.59	28.00	5.38
TRIVIUM	72	5.867	19.69	49.59	5.59	28.00	0.55	19.69	49.59	5.59	28.01	0.55

Table A5. Implementations results of the cryptographic algorithms on the Arduino Nano board.

Algorithm	RAM Memory [B]	ROM/FLASH Memory [kB]	Encryption Latency [µs/B]	Encryption Throughput [kB/s]	Encryption Electric Current [mA]	Encryption Electric Power [mW]	Encryption Energy [µJ/B]	Decryption Latency [µs/B]	Decryption Throughput [kB/µs]	Decryption Electric Current [mA]	Decryption Electric Power [mW]	Decryption Energy [µJ/B]
PRESENT	345	2.342	2214.29	0.44	10.14	50.69	112.24	3460.26	0.28	10.08	50.41	174.43
SPECK	642	2.992	9.71	100.55	10.51	52.54	0.51	10.09	96.79	10.51	52.54	0.53
Piccolo80	537	1.566	41.36	23.61	10.35	51.74	2.14	54.45	17.93	10.52	52.58	2.86
CRAFT	411	2.195	461.18	2.12	10.47	52.33	24.13	462.48	2.11	10.46	52.29	24.18
Hummingbird-2	487	3.600	225.03	4.34	10.15	50.74	11.42	238.18	4.10	10.21	51.05	12.16
ASCON128	609	7.959	263.12	3.71	10.54	52.69	13.86	263.31	3.71	10.54	52.69	13.87
ACORN128	417	5.961	2990.18	0.33	10.16	50.79	151.87	2290.18	0.33	10.16	50.80	116.34
Lizard	352	3.523	3009.78	0.32	10.35	51.74	155.72	3009.89	0.32	10.36	51.79	155.88
Fruit-80	344	2.783	1292.06	0.76	10.23	51.14	66.08	1292.06	0.76	10.24	51.19	66.14
TRIVIUM	369	2.906	594.09	1.64	10.15	50.75	30.15	594.09	1.64	10.15	50.75	30.15

References

1. Yu, J.-Y.; Lee, E.; Oh, S.-R.; Seo, Y.-D.; Kim, Y.-G. A Survey on Security Requirements for WSNs: Focusing on the Characteristics Related to Security. *IEEE Access* **2020**, *8*, 45304–45324. [CrossRef]
2. Tomić, I.; McCann, J.A. A Survey of Potential Security Issues in Existing Wireless Sensor Network Protocols. *IEEE Internet Things J.* **2017**, *4*, 1910–1923. [CrossRef]
3. Sun, P.; Shen, S.; Wan, Y.; Wu, Z.; Fang, Z.; Gao, X.-Z. A Survey of IoT Privacy Security: Architecture, Technology, Challenges, and Trends. *IEEE Internet Things J.* **2024**, *11*, 34567–34591. [CrossRef]
4. Adil, M.; Menon, V.G.; Balasubramanian, V.; Alotaibi, S.R.; Song, H.; Jin, Z. Survey: Self-Empowered Wireless Sensor Networks Security Taxonomy, Challenges, and Future Research Directions. *IEEE Sens. J.* **2023**, *23*, 20519–20535. [CrossRef]
5. Alawad, F.; Kraemer, F.A. Value of Information in Wireless Sensor Network Applications and the IoT: A Review. *IEEE Sens. J.* **2022**, *22*, 9228–9245. [CrossRef]
6. Vieira, E.; Almeida, J.; Ferreira, J.; Bartolomeu, P.C. Enabling Seamless Data Security, Consensus, and Trading in Vehicular Networks. *IEEE Trans. Intell. Veh.* **2024**, 1–12. [CrossRef]
7. Tahir, B.; Jolfaei, A.; Tariq, M. A Novel Experience-Driven and Federated Intelligent Threat-Defense Framework in IoMT. *IEEE J. Biomed. Health Inform.* **2023**, 1–8. [CrossRef]
8. Islam, K.; Shen, W.; Wang, X. Wireless Sensor Network Reliability and Security in Factory Automation: A Survey. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2012**, *42*, 1243–1256. [CrossRef]
9. Nguyen, H.A.D.; Ha, Q.P. Wireless Sensor Network Dependable Monitoring for Urban Air Quality. *IEEE Access* **2022**, *10*, 40051–40062. [CrossRef]
10. Surekha, S.; Rahman, M.Z.U. Cognitive Energy-Aware Spectrum Sensing with Improved Throughput for Medical Sensor Networks. *IEEE Sens. Lett.* **2022**, *6*, 5500904. [CrossRef]
11. Lin, C.-C.; Yen, T.-C.; Chen, Y.-S. Multiple Time-sensitive Inferences Scheduling on Energy-harvesting IoT Devices. In Proceedings of the RACS '23: Proceedings of the 2023 International Conference on Research in Adaptive and Convergent Systems, Gdansk, Poland, 6–10 August 2023. [CrossRef]
12. Hernández, N.; Almeida, F.; Blanco, V. Performance and energy efficiency: Quantization of models for IoT devices. *Res. Sq.* **2023**. [CrossRef]
13. Weng, D. Performance and Energy Evaluation of Lightweight Cryptography for Small IoT Devices. In Proceedings of the IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 12–14 October 2023. [CrossRef]
14. Wajgi, D.; Tembhurne, J.V.; Wajgi, R.; Jain, T. Communication in IoT Devices. In *Modern Approaches in IoT and Machine Learning for Cyber Security*; Internet of Things Book Series; Springer: Cham, Switzerland, 2023; pp. 21–44. [CrossRef]
15. Yu, Z.; Gao, H.; Cong, X.; Wu, N.; Song, H.H. A Survey on Cyber–Physical Systems Security. *IEEE Internet Things J.* **2023**, *10*, 21670–21686. [CrossRef]
16. Saba, S.J.; Al-Nuaimi, B.T.; Suhail, R.A. A review of traditional, lightweight and ultra-lightweight cryptography techniques for IoT security environment. *AIP Conf. Proc.* **2023**, *2475*, 070001. [CrossRef]
17. Kamaldeep; Dutta, M.; Granjal, J. Towards a Secure Internet of Things: A Comprehensive Study of Second Line Defense Mechanisms. *IEEE Access* **2020**, *8*, 127272–127312. [CrossRef]
18. Noor, M.B.M.; Hassan, W.H. Current research on Internet of Things (IoT) security: A survey. *Comput. Netw.* **2019**, *148*, 283–294. [CrossRef]
19. Murtaza, G.; Iqbal, F.; Altaf, A.; Rasheed, A. Techniques for Resource-Efficient, Lightweight Cryptography in IoT Devices for Smart Environment. In Proceedings of the 2023 Sixth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU), Riyadh, Saudi Arabia, 14–15 March 2023; pp. 223–228. [CrossRef]

20. Kapalova, N.; Algazy, K.; Haumen, A. Development of a new lightweight encryption algorithm. *East.-Eur. J. Enterp. Technol.* **2023**, *3*, 6–19. [CrossRef]
21. Abubakar, I.; Sani, M.; Surajo, Y. A New Lightweight Cryptographic Cipher for Detection and Prevention of Replay Attacks in Wireless Sensor Networks. *Int. J. Sci. Glob. Sustain.* **2024**, *10*, 33–40. [CrossRef]
22. Khan, M.N.; Rao, A.; Camtepe, S. Lightweight Cryptographic Protocols for IoT Constrained Devices: A Survey. *IEEE Internet Things J.* **2020**, *8*, 4132–4156. [CrossRef]
23. Abdelli, A.; El hadj Youssef, W.; Kharroubi, F.; Khriji, L.; Machhout, M. A novel enhanced chaos based present lightweight cipher scheme. *Phys. Scr.* **2024**, *99*, 016004. [CrossRef]
24. Hughes, L.E. Basic Cryptography: Symmetric Key Encryption. In *Pro Active Directory Certificate Services*; Apress: Berkeley, CA, USA, 2022. [CrossRef]
25. Dutta, I.K.; Ghosh, B.; Bayoumi, M. Lightweight Cryptography for Internet of Insecure Things: A Survey. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 0475–0481. [CrossRef]
26. Dhanda, S.S.; Singh, B.; Jindal, P. Lightweight Cryptography: A Solution to Secure IoT. *Wirel. Pers. Commun.* **2020**, *112*, 1947–1980. [CrossRef]
27. Gunathilake, N.A.; Al-Dubai, A.; Buchana, W.J. Recent Advances and Trends in Lightweight Cryptography for IoT Security. In Proceedings of the 2020 16th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2–6 November 2020; pp. 1–5. [CrossRef]
28. Paar, C.; Pelzl, J. *Understanding Cryptography*; Springer: Berlin/Heidelberg, Germany, 2010. [CrossRef]
29. Suwais, K. Stream Cipher Based on Game Theory and DNA Coding. *Intell. Autom. Soft Comput.* **2022**, *33*, 1815–1834. [CrossRef]
30. Hatzivasilis, G.; Fysarakis, K.; Papaefstathiou, I.; Manifavas, C. A review of lightweight block ciphers. *J. Cryptogr. Eng.* **2017**, *8*, 141–184. [CrossRef]
31. Jassim, S.A.; Farhan, A.K. A Survey on Stream Ciphers for Constrained Environments. In Proceedings of the 2021 1st Babylon International Conference on Information Technology and Science (BICITS), Babil, Iraq, 28–29 April 2021; pp. 228–233. [CrossRef]
32. Meng, T.X.; Buchanan, W.J. Lightweight Cryptographic Algorithms on Resource-Constrained Devices. *Comput. Sci. Math.* **2020**, 2020090302. [CrossRef]
33. Kureshi, R.R.; Mishra, B.K. A Comparative Study of Data Encryption Techniques for Data Security in the IoT Device. In *Internet of Things and Its Applications*; Lecture Notes in Electrical Engineering Book Series; Springer: Berlin/Heidelberg, Germany, 2022; pp. 451–460. [CrossRef]
34. Sevin, A.; Mohammed, A.A.O. A survey on software implementation of lightweight block ciphers for IoT devices. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *14*, 1801–1815. [CrossRef]
35. Jiao, L.; Hao, Y.; Feng, D. Stream cipher designs: A review. *Sci. China Inf. Sci.* **2020**, *63*, 131101. [CrossRef]
36. Thakor, V.A.; Razzaque, M.A.; Khandaker, M.R.A. Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities. *IEEE Access* **2021**, *9*, 28177–28193. [CrossRef]
37. Dorobantu, O.G.; Apostol, A.-G.; Datcu, O. The poly-alphabetic substitution ciphers—A viable solution for IoT applications? In Proceedings of the 2022 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 10–11 November 2022; pp. 1–4. [CrossRef]
38. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.; Seurin, Y.; Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware and Embedded Systems—CHES 2007*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 450–466. [CrossRef]
39. Jeong, K.; Lee, Y.; Sung, J.; Hong, S. Improved differential fault analysis on PRESENT-80/128. *Int. J. Comput. Math.* **2013**, *90*, 2553–2563. [CrossRef]
40. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK Families of Lightweight Block Ciphers. ePrint IACR. 2013. Available online: https://eprint.iacr.org/2013/404 (accessed on 16 February 2024).
41. Abed, F.; List, E.; Lucks, S.; Wenzel, J. Cryptanalysis of the Speck Family of Block Ciphers. Cryptology ePrint Archive (eprint.iacr.org). 2013. Available online: https://eprint.iacr.org/2013/568 (accessed on 18 April 2024).
42. Shibutani, K.; Isobe, T.; Hiwatari, H.; Mitsuda, A.; Akishita, T.; Shirai, T. Piccolo: An Ultra-Lightweight Blockcipher. Available online: https://www.iacr.org/archive/ches2011/69170343/69170343.pdf (accessed on 9 February 2024).
43. Liu, Y.; Liu, C.; Liu, Z.; Li, W.; Wang, Q.; Gu, D. Improved meet-in-the-middle attacks on reduced-round Piccolo. *Sci. China Inf. Sci.* **2017**, *61*, 032108. [CrossRef]
44. Wang, Y.; Wu, W.; Yu, X. Biclique Cryptanalysis of Reduced-Round Piccolo Block Cipher. In *Information Security Practice and Experience*; Lecture Notes in Computer Science Book Series; Springer: Berlin/Heidelberg, Germany, 2012; pp. 337–352. [CrossRef]
45. Beierle, C.; Leander, G.; Moradi, A.; Rasoolzadeh, S. CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks. *IACR Trans. Symmetric Cryptol.* **2019**, *2019*, 5–45. [CrossRef]
46. Pang, K.; Abdul-Latip, S.F. Key-dependent side-channel cube attack on CRAFT. *ETRI J.* **2021**, *43*, 344–356. [CrossRef]

47. Engels, D.; Saarinen, M.-J.; Schweitzer, P.; Smith, E. The Hummingbird-2 Lightweight Authenticated Encryption Algorithm. Available online: https://eprint.iacr.org/2011/126.pdf (accessed on 3 June 2024).
48. Saarinen, M.-J.O. Related-key Attacks Against Full Hummingbird-2. Cryptology ePrint Archive (eprint.iacr.org). 2013. Available online: https://eprint.iacr.org/2013/070 (accessed on 3 June 2024).
49. Schläffer, C.D.; Eichlseder, M.; Mendel, F.M. Ascon—Authenticated Encryption and Hashing. Available online: https://ascon.iaik.tugraz.at/ (accessed on 9 February 2024).
50. Computer Security Division of Information Technology Laboratory. Announcing Lightweight Cryptography Selection | CSRC.CSRC | NIST. 6 February 2023. Available online: https://csrc.nist.rip/News/2023/lightweight-cryptography-nist-selects-ascon (accessed on 12 February 2024).
51. Weissbart, L.; Picek, S. Lightweight but Not Easy: Side-Channel Analysis of the Ascon Authenticated Cipher on a 32-Bit Microcontroller. Cryptology ePrint Archive (eprint.iacr.org). 2023. Available online: https://eprint.iacr.org/2023/1598 (accessed on 18 April 2024).
52. Wu, H. ACORN: A Lightweight Authenticated Cipher (v3). 2016. Available online: https://competitions.cr.yp.to/round3/acornv3.pdf (accessed on 18 April 2024).
53. Zhang, X.; Lin, D. Cryptanalysis of Acorn in Nonce-Reuse Setting. In *Information Security and Cryptology*; Lecture Notes in Computer Science Book Series; Springer: Berlin/Heidelberg, Germany, 2018; pp. 342–361. [CrossRef]
54. Hamann, M.; Krause, M.; Meier, W. LIZARD–A Lightweight Stream Cipher for Power-constrained Devices. *IACR Trans. Symmetric Cryptol.* **2017**, *2017*, 45–79. [CrossRef]
55. Baksi, A.; Kumar, S.; Sarkar, S. A New Approach for Side Channel Analysis on Stream Ciphers and Related Constructions. *IEEE Trans. Comput.* **2022**, *71*, 2527–2537. [CrossRef]
56. Ghafari, V.A.; Hu, H. Fruit-80: A Secure Ultra-Lightweight Stream Cipher for Constrained Environments. *Entropy* **2018**, *20*, 180. [CrossRef] [PubMed]
57. Todo, Y.; Meier, W.; Aoki, K. On the Data Limitation of Small-State Stream Ciphers: Correlation Attacks on Fruit-80 and Plantlet. In *Selected Areas in Cryptography—SAC 2019*; Lecture Notes in Computer Science Book Series; Springer: Berlin/Heidelberg, Germany, 2020; pp. 365–392. [CrossRef]
58. De Cannière, C.; Preneel, B. Trivium Specifications. Available online: https://cr.yp.to/streamciphers/trivium/desc.pdf (accessed on 5 April 2024).
59. Potestad-Ordóñez, F.E.; Valencia-Barrero, M.; Baena-Oliva, C.; Parra-Fernández, P.; Jiménez-Fernández, C.J. Breaking Trivium Stream Cipher Implemented in ASIC Using Experimental Attacks and DFA. *Sensors* **2020**, *20*, 6909. [CrossRef] [PubMed]
60. Sadhu, A.K.R.; Reddy, A.K.y. A Comparative Analysis of Lightweight Cryptographic Protocols for Enhanced Communication Security in Resource-Constrained Internet of Things (IoT) Environments. *Emerg. Res.* **2022**, *2*, 121–142.
61. José, M. A Comparative Study on the Performance of Security Mechanisms in Internet of Things Devices. *Engineering* **2023**, 2023060529. [CrossRef]
62. Kumar, S.; Kumar, D.; Dangi, R.; Choudhary, G.; Dragoni, N.; You, I. A Review of Lightweight Security and Privacy for Resource-Constrained IoT Devices. *CMC* **2024**, *78*, 31–63. [CrossRef]
63. Madushan, H.; Salam, I.; Alawatugoda, J. A Review of the NIST Lightweight Cryptography Finalists and Their Fault Analyses. *Electronics* **2022**, *11*, 4199. [CrossRef]
64. Eisenbarth, T.; Kumar, S.; Paar, C.; Poschmann, A.; Uhsadel, L. A Survey of Lightweight-Cryptography Implementations. *IEEE Des. Test. Comput.* **2007**, *24*, 522–533. [CrossRef]
65. Suryateja, P.S.; Rao, K.V. A Survey on Lightweight Cryptographic Algorithms in IoT. *Cybern. Inf. Technol.* **2024**, *24*, 21–34. [CrossRef]
66. El-hajj, M.; Mousawi, H.; Fadlallah, A. Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platform. *Future Internet* **2023**, *15*, 54. [CrossRef]
67. Kumar, G.S.; Babu, D.A.R. Comparative Analysis of Light Weight algorithms based on Encryption in Cloud Data Security Environment. In Proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 10–12 June 2020.
68. Lee, D.; Kim, D.-C.; Kwon, D.; Kim, H. Efficient Hardware Implementation of the Lightweight Block Encryption Algorithm LEA. *Sensors* **2014**, *14*, 975–994. [CrossRef] [PubMed]
69. Naseer, M.; Tariq, S.; Riaz, N.; Ahmed, N.; Hussain, M. S-box Security Analysis of NIST Lightweight Cryptography Candidates: A Critical Empirical Study. *arXiv* **2024**, arXiv:2404.06094. [CrossRef]
70. Caraveo-Cacep, M.A.; Vázquez-Medina, R.; Zavala, A.H. A survey on low-cost development boards for applying cryptography in IoT systems. *Internet Things* **2023**, *22*, 100743. [CrossRef]
71. Plaza, P.; Sancristobal, E.; Carro, G.; Castro, M.; Ruiz, E.R. Wireless Development Boards to Connect the World. In *Online Engineering & Internet of Things*; Lecture Notes in Networks and Systems Book Series; Springer: Berlin/Heidelberg, Germany, 2018; pp. 19–27. [CrossRef]

72. Singh, D.; Sandhu, A.; Thakur, A.; Priyank, N. An Overview of IoT Hardware Development Platforms. *Int. J. Emerg. Technol.* **2020**, *11*, 155–163.
73. Arduino Cryptography Library. rweather.github.io. Available online: https://rweather.github.io/arduinolibs/crypto.html (accessed on 19 April 2024).
74. Tonkovic, P. Pepton21/Present-Cipher. GitHub. 15 October 2023. Available online: https://github.com/Pepton21/present-cipher (accessed on 19 April 2024).
75. Jovanovic, P. Daeinar/Piccolo. GitHub. 2 March 2022. Available online: https://github.com/Daeinar/piccolo (accessed on 19 April 2024).
76. Chandra, S.; Paira, S.; Alam, S.S.; Sanyal, G. A comparative survey of Symmetric and Asymmetric Key Cryptography. In Proceedings of the 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE), Hosur, India, 17–18 November 2014; pp. 83–93. [CrossRef]
77. Zhang, Q. An Overview and Analysis of Hybrid Encryption: The Combination of Symmetric Encryption and Asymmetric Encryption. In Proceedings of the 2021 2nd International Conference on Computing and Data Science (CDS), Stanford, CA, USA, 28–29 January 2021; pp. 616–622. [CrossRef]
78. Khashan, O.A. Hybrid Lightweight Proxy Re-Encryption Scheme for Secure Fog-to-Things Environment. *IEEE Access* **2020**, *8*, 66878–66887. [CrossRef]
79. Ning, J.; Cao, Z.; Dong, X.; Liang, K.; Ma, H.; Wei, L. Auditable σ -Time Outsourced Attribute-Based Encryption for Access Control in Cloud Computing. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 94–105. [CrossRef]
80. Deng, H.; Qin, Z.; Wu, Q.; Guan, Z.; Deng, R.H.; Wang, Y.; Zhou, Y. Identity-Based Encryption Transformation for Flexible Sharing of Encrypted Data in Public Cloud. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3168–3180. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.