Article

Not peer-reviewed version

A Survey of Efficient Lightweight Cryptography for Power-Constrained Microcontrollers

Jesús Alejandro Soto-Cruz , Érica Cecilia Ruiz-Ibarra * , Javier Vázquez-Castillo , Adolfo Espinoza-Ruiz , Alejandro Castillo-Atoche , Joaquin Mass-Sanchez

Posted Date: 31 October 2024

doi: 10.20944/preprints202410.2545.v1

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

 doi:10.20944/preprints202410.2545.v1

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 Joaquín Mass-Sánchez [1,]

[1] Departamento de Ingeniería Eléctrica y Electrónica, Instituto Tecnológico de Sonora, ITSON, Cd. Obregón, Son. México. 85130.
[2] Departamento de Redes e Informática, Universidad Autónoma del Estado de Quintana Roo (UQRoo), Chetumal, Quintana Roo, México. 77019.
[3] Departamento de Mecatrónica, Universidad Autónoma de Yucatán, Mérida, Yucatán, México, 97000.
* 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 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

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. 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 [11]. 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 their interconnected nature and the limited resources of IoT devices [12,13]. Figure 1 shows 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 since it facilitates secure communication between devices, in addition to data protection and user privacy [14]. Despite this, traditional cryptography is not a viable mechanism for IoT devices since it is highly resource-demanding [15]. As an alternative to traditional cryptography, and considering the low amount of resources of these devices, a new branch of cryptography, known as lightweight cryptography, has been introduced and designed specifically for this type of environment [16].

 doi:10.20944/preprints202410.2545.v1

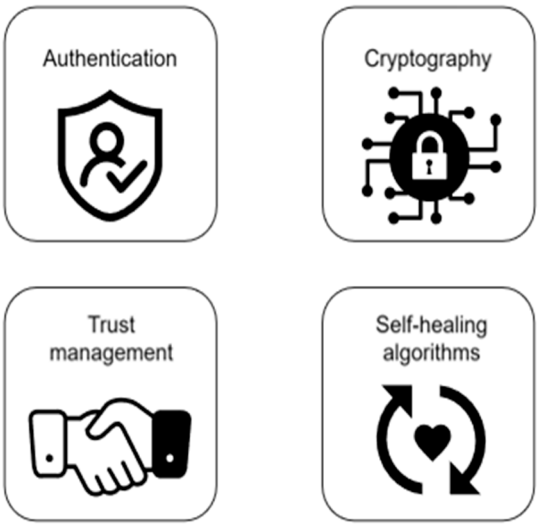


Figure 1. Security Mechanisms used in IoT.

The feasibility of using lightweight cryptography to secure IoT devices with limited power and computational resources has recently started to be investigated. 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 done 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 efficient implementation of secure monitoring systems. The experimental results obtained as a product of this work are of great value in determining the performance of the different lightweight algorithms reviewed and their best use on low-power microcontrollers, depending on 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. Finally, the conclusions for this work are presented in Section 6.

2. Related Work

Within cryptography itself, there are two types of methods used to protect transmitted information: symmetric and asymmetric encryption [17]. 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) is used, where a key is used to encrypt a message that can only be decrypted using the other key. The work presented in this paper focuses only on the first method, symmetric encryption; thus, more will be said about this type of method. 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.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 October 2024 doi:10.20944/preprints202410.2545.v1

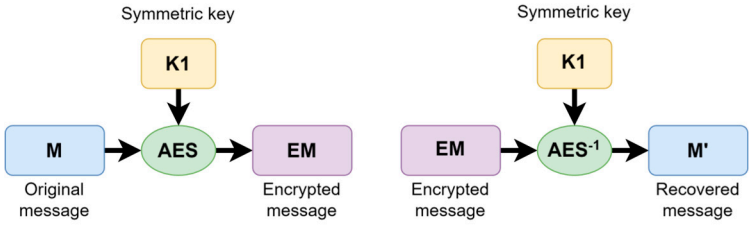


Figure 2. Symmetric encryption process diagram.

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 [18], whereas the stream cipher generates a pseudo-random vector called "stream key" and encrypts the message with it using a XOR operation [19]. Figure 3 describes the encryption and decryption process used in a block cipher, and Figure 4 illustrates the process 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 to decrypt, but in a stream cipher, the same function is used for encryption as well as decryption since the inverse XOR operation is the same operation.

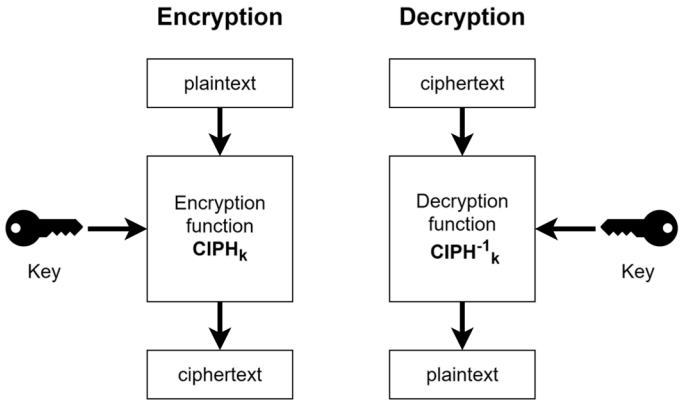


Figure 3. Encryption and decryption process using a block cipher algorithm.

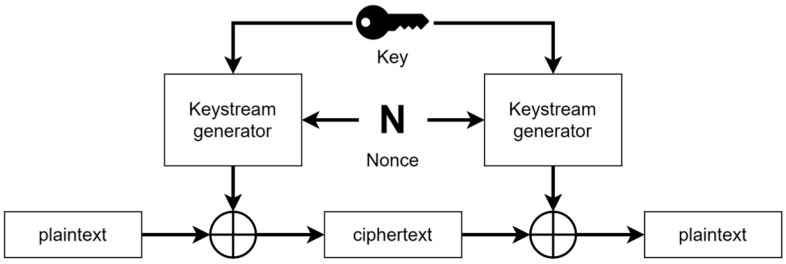


Figure 4. Encryption and decryption process using a stream cipher algorithm.

Symmetric cryptography has recently started to be studied for its use in IoT; Hatzivasilis *et al.* [20] 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. & Farhan A. [21] conducted a survey focused on 27 lightweight stream ciphers, evaluating their design patterns, key size, vulnerabilities shown in

 doi:10.20944/preprints202410.2545.v1

previously performed studies, and their adaptability for constrained devices. Both Hatzivasilis *et al.* [20] and Jassim S. & Farhan A. [21] 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.* [22] implemented and evaluated both block and stream cipher algorithms, focusing primarily 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 as 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 done on devices with a substantially high amount 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 & B. Mishra [23] performs 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 to those commonly used in IoT, therefore, this study focuses on traditional encryption algorithms for IoT.

A. Sevin & A. Osman [24] present a study of 50 lightweight block ciphers implemented in software for IoT to evaluate their performance in regard to execution time, latency, ROM, and RAM usage, as well as 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 in 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, and point out the need to evaluate stream ciphers.

L. Jiao *et al.* [25] review the history of stream ciphers and how they have been developed over the years in such a way that these can be efficient on hardware implementations without neglecting an adequate level of security even with a reduced amount of resources. This review classifies stream ciphers into groups according to their design principles, then uses these 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 since no variables of interest are shown that can be useful to developers looking for specific needs.

V. Thakor *et al.* [26], similarly to [20], conducted a survey of 50 popular block ciphers used in IoT devices, evaluating them regarding their design features, performance, and security measures. Security issues are also addressed taking into consideration the constrained resources of IoT devices, stressing the importance of lightweight cryptography for secure data transfer. The algorithms shown are implemented on microcontrollers from NXP, AVR, and ARM platforms. The algorithms are evaluated based on their structure design, assisting their comparison and helping developers make decisions depending on their needs. Despite briefly mentioning stream cipher algorithms, this paper does not evaluate algorithms of this type since 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 the available options. One of the branches that should be further studied involves stream ciphers, given they are a good alternative for data protection and most of the papers mentioned in this section does not include them.

 doi:10.20944/preprints202410.2545.v1

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 [22]. Traditional encryption algorithms used in computers, such as the AES standard, are valuable tools 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 [27]. Therefore, it is highly advisable to use lightweight encryption algorithms that use the least amount of computational resources possible and guarantee a strong level security as a security mechanism for IoT. The following algorithms were evaluated in this study.

3.1. Lightweight Block Ciphers

- **PRESENT** [28]: This algorithm is designed for environments where the amount of resources is limited, 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 [29].
- **SPECK** [30]: 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 [31]. Despite this, the algorithm is considered a great alternative due to its high performance in IoT applications.
- **Piccolo** [32]: This 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 [33,34]. 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** [35]: 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 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 [36]. Given that this algorithm was developed recently, further research is needed to mitigate the possible attacks that this cipher is susceptible to.
- **Hummingbird-2** [37]: It is a lightweight cryptography algorithm designed for resource-constrained devices that uses a 128-bit secret key and a 64-bit initialization 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 [38].

Table 1 summarizes the features of the lightweight block cipher 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

 doi:10.20944/preprints202410.2545.v1

3.2. Lightweight Stream Ciphers

- **ASCON** [39]: 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 [40]. It was designed specifically to withstand various cryptographic attacks; however, additional measurements are required to ensure its security in practical implementations [41].
- **ACORN** [42]: 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 [43], so further studies are necessary to determine whether the algorithm is appropriate to ensure the security of IoT devices.
- **Lizard** [44]: 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 [45].
- **Fruit-80** [46]**:** 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 significant optimizations for hardware implementation. Several security analyses have detected potential vulnerabilities when it falls victim to correlation attacks [47]. Despite this, Fruit-80 offers strong resistance against known attacks and is a viable option for IoT due to its efficiency.
- **TRIVIUM** [48]: This algorithm is designed to operate efficiently on both software and hardware. 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 this reason TRIVIUM 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 [49]. 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

4. Lightweight Cryptography Algorithms Analysis

In this section, the methodology used to evaluate the symmetric encryption algorithms studied is described. Alongside with 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.

 doi:10.20944/preprints202410.2545.v1

4.1. State-of-the-Art Research and Opportunity Ideas Identification

Relevant information regarding the current overview of security for embedded systems used in IoT was collected in order to have an overview of the topic. Subsequently, opportunities for improvements in the security of 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 [20–26]. Only 10 algorithms were selected due to time considerations for this research. The selected algorithms were mentioned in section 3 above.

4.3. Hardware and Software Selection

Development boards were chosen among the most popular for IoT projects, according to the State-of-the-art reviewed [50–52], which 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 3.

Table 3. Specifications for the development boards used.

Development board	**ESP32 Dev Module**	**NodeMCU ESP8266**	**MSP430 Launchpad**
Data width	32-bit	32-bit	16-bit
Clock [MHz]	240	160	16
RAM [kB]	520	64	0.5
ROM [kB]	448		
FLASH [kB]		512	16
Pins	30 - 36	30	24
Supply voltage[V]	5	5	3.6
I/O voltage [V]	3.3	3.3	3.6
Supply voltage[V]	5	5	3.6

 doi:10.20944/preprints202410.2545.v1

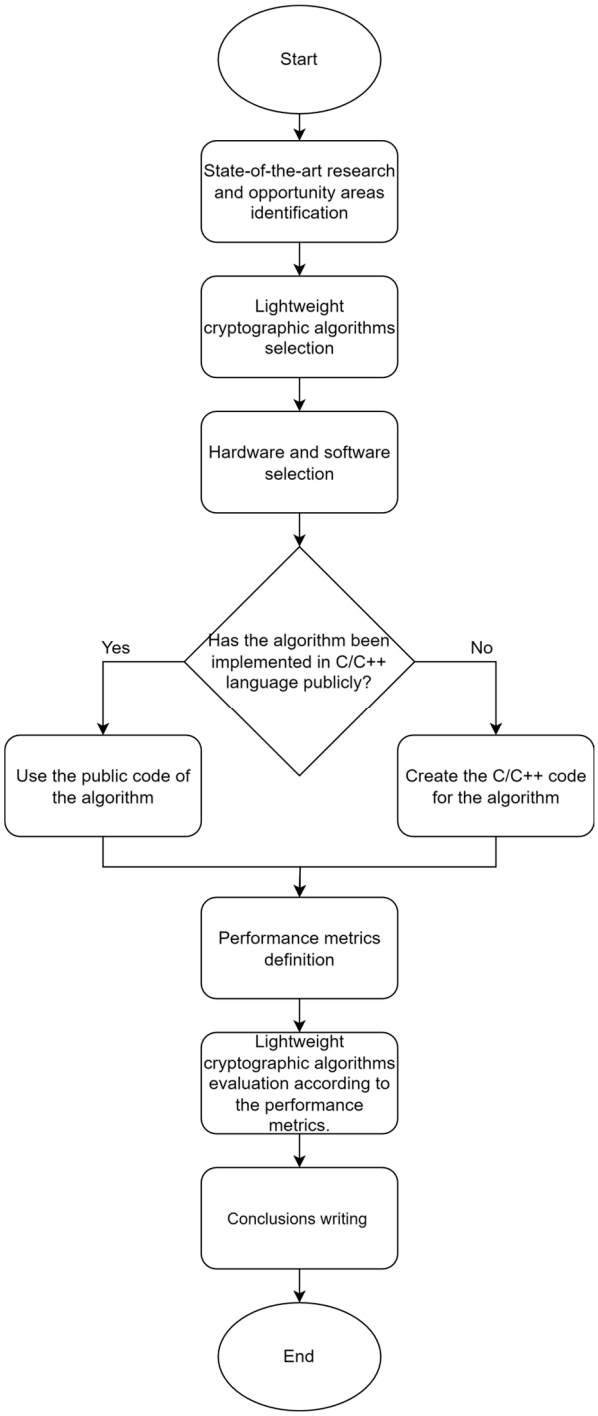


Figure 5. Flowchart of the work methodology used in this research.

4.4. Code Implementation

All the code implementations used to evaluate the lightweight cryptographic algorithms were programmed 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.

ASCON and SPECK algorithms were implemented using the Arduino Crypto library developed by Rhys Wheatherley [53], which provides traditional and lightweight cryptography algorithms. PRESENT and Piccolo algorithms were implemented based on code published on GitHub [54,55] that was modified for its use in development boards. The remaining algorithms, CRAFT, Hummingbird-

 doi:10.20944/preprints202410.2545.v1

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

Performance metrics were defined to evaluate the previously mentioned algorithms. 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 was run 5000 times, and the time required to carry out these executions was measured. The time required per byte was calculated using the equation 1:

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

- 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. Then, using this time and the data size of the processed message, the metric was obtained using equation 2:

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

- Current: The average value of electrical current each development board required for encryption/decryption. Each of the algorithms was 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 Figure 6.
- Power: The average transfer rate of electrical energy required to encrypt/decrypt by each of the algorithms. This parameter was calculated using the current measured mentioned above and the 5 V supply voltage value as shown in equation 3:

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

- 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:

$$Energy\ \left[\frac{\text{J}}{\text{byte}}\right] = Power\ [\text{W}] * latency\ \left[\frac{\text{s}}{\text{bytes}}\right] \tag{4}$$

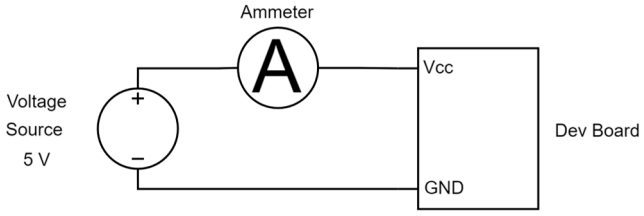


Figure 6. Electric circuit used to measure current in the development boards.

 doi:10.20944/preprints202410.2545.v1

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 cards previously mentioned. Different messages were encrypted to verify the correct operation of the algorithms, 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 its correct operation. The development boards were used to execute the symmetric cryptographic algorithms and for evaluation according to the performance metrics. The development boards were connected to a computer to visualize the correct operation of the algorithms and the results. These were programmed using the PlatformIO extension available in the Visual Studio Code IDE, supported by the Arduino framework for easy implementation. It is worth mentioning that the same code corresponding to each algorithm was used to program each of the development boards mentioned.

4.7. Conclusions Writing

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

5. Results

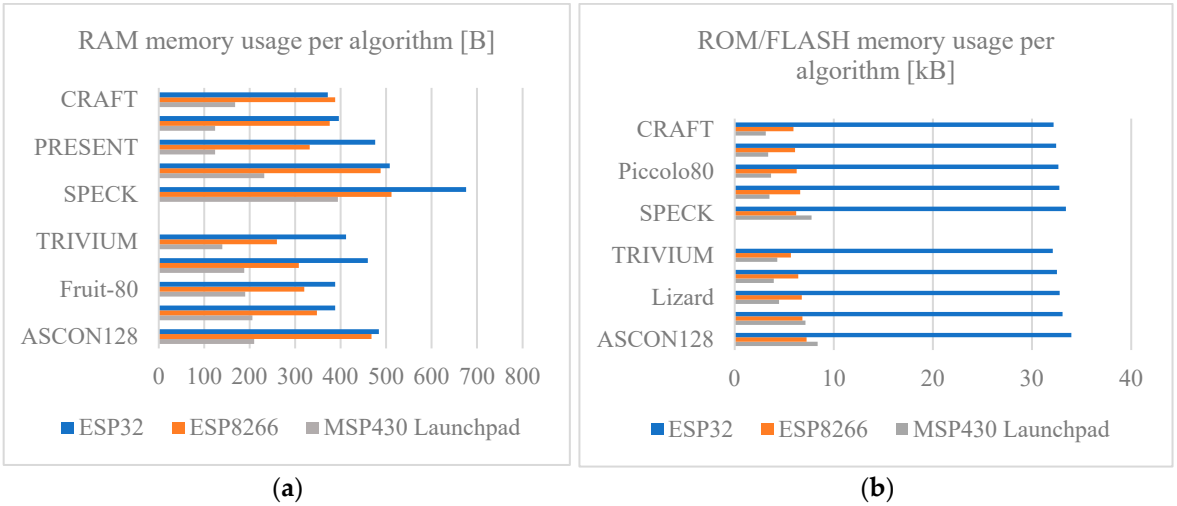
This section shows the results obtained from the implementation of each of the symmetric encryption algorithms studied as well as their analysis, evaluation and comparison. The results are shown as graphs to facilitate comparison.

5.1. Memory Usage

Figure 7 shows the amount of RAM memory required by each symmetric algorithm, both RAM and ROM/FLASH. Focusing on Figure 7a, the results obtained reveal significant variations among the algorithms and even a greater variation among boards used. It can be easily noted that the development board that requires the largest amount of RAM memory to implement the algorithms is the ESP32 board. On the other hand, the development board that requires the least amount of RAM memory is the MSP430 Launchpad. This huge difference mainly occurs due to the fact that the compilation performed by the platformIO is not well optimized for the ESP32 board since it has a considerable amount of RAM. For the MSP430 Launchpad case, having such a low amount of resources, the code compilation is better suited to avoid wasting memory. Comparing the block and stream cipher algorithms, it is noted that there is no apparent relationship among them in RAM usage, as they have different values among each other. The differences in RAM memory usage between the development boards are mainly due to how well-optimized the compilation for each of them is as well as their architecture, given that the ESP family boards use a 32-bit architecture while the MSP430 Launchpad uses a 16-bit architecture.

Similarly, to the previous point, Figure 7b shows the ROM/FLASH memory usage required to implement each of the cryptographic algorithms studied. The variations in ROM/FLASH memory usage between algorithms are minimal. However, the variation between boards is very noticeable; the ESP32 board uses significantly more memory compared to the ESP8266 and MSP430 Launchpad boards. This is highly related to the RAM memory usage previously observed; therefore, it is assumed that the higher ROM/FLASH memory consumption on the ESP32 board comes from the poor optimization of the PlatformIO compilation for this board since the ESP8266 and MSP430 Launchpad boards have a similar memory usage.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 October 2024 doi:10.20944/preprints202410.2545.v1



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

Figure 7. Memory usage per algorithm. **(a)** RAM memory; **(b)** ROM/FLASH memory.

5.2. Encryption and Decryption Latency

Figure 8–10 show the algorithm latency for processing a single byte of data for both encryption and decryption measured on the ESP32, ESP8266, and MSP430 Launchpad boards, respectively. The encryption and decryption latency times are consistent with each other for most of the algorithms analyzed; that is expected since both operations are inverse-proportional between them.

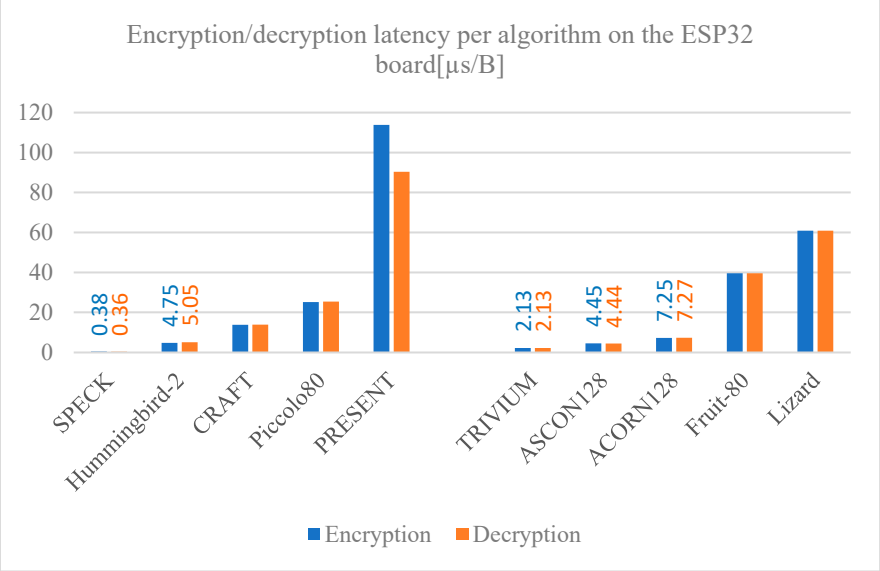


Figure 8. Encryption/decryption latency per algorithm on the ESP32 board.

Significant differences are noted when comparing latency among the algorithms; the PRESENT block cipher stands out as the lowest, requiring more time to process one byte of data compared to the other algorithms. On the other hand, the SPECK algorithm has the lowest processing. Comparing the latency time among the development boards, the MSP430 Launchpad has a remarkably higher latency compared to those of the ESP family across all the algorithms, where the times are slightly higher on ESP8266 compared to ESP32. The reason for this is the clock speed of the boards used; the MSP430 Launchpad has an 18 MHz clock, which is considered slow when compared to those on the ESP8266 and ESP32 boards, which have an 80 MHz and 240 MHz clock speed, respectively.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 October 2024 doi:10.20944/preprints202410.2545.v1

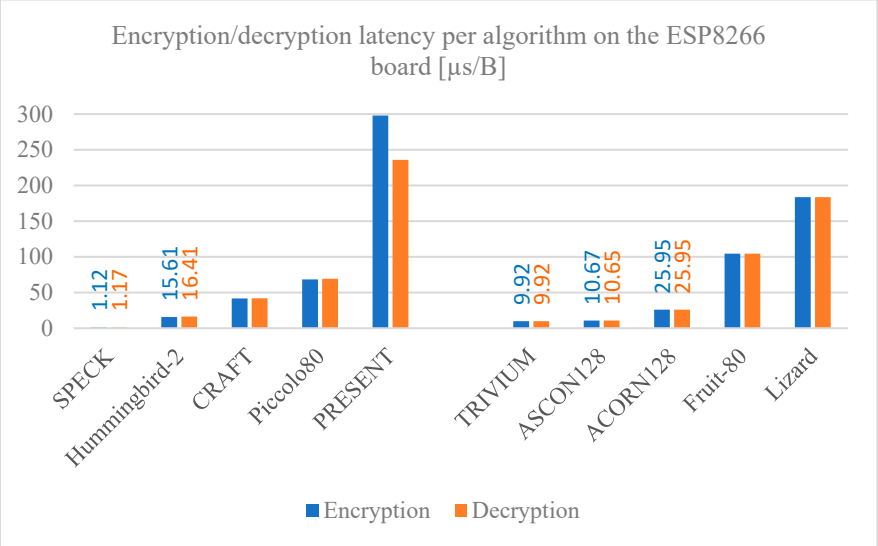


Figure 9. Encryption/decryption latency per algorithm on the ESP8266 board.

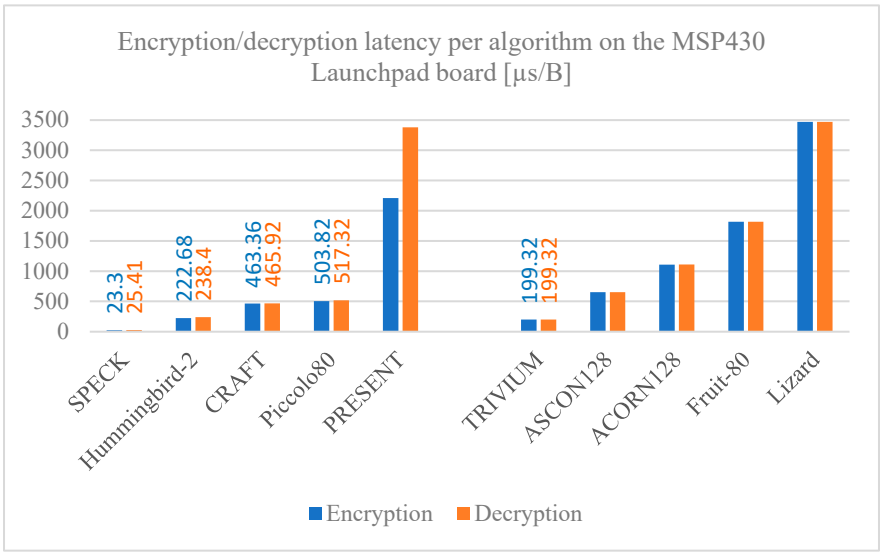


Figure 10. Encryption/decryption latency per algorithm on the MSP430 Launchpad board.

5.3. Throughput for Encryption and Decryption

Figure 11–13 show the throughput for encryption and decryption of the selected algorithms implemented on the ESP32, ESP8266, and MSP430 Launchpad boards. Comparing these results with previous ones, it can be seen that latency and throughput are inversely proportional to each other, expectedly, since if less time is required to process one byte of data, more bytes can be processed in a given time. The SPECK algorithm stands out with the highest performance in any of the boards, which must be attributed to its design, followed closely by the TRIUVIUM stream cipher algorithm. Otherwise, the PRESENT, Lizard, and Fruit-80 algorithms have the worst throughput.

 doi:10.20944/preprints202410.2545.v1

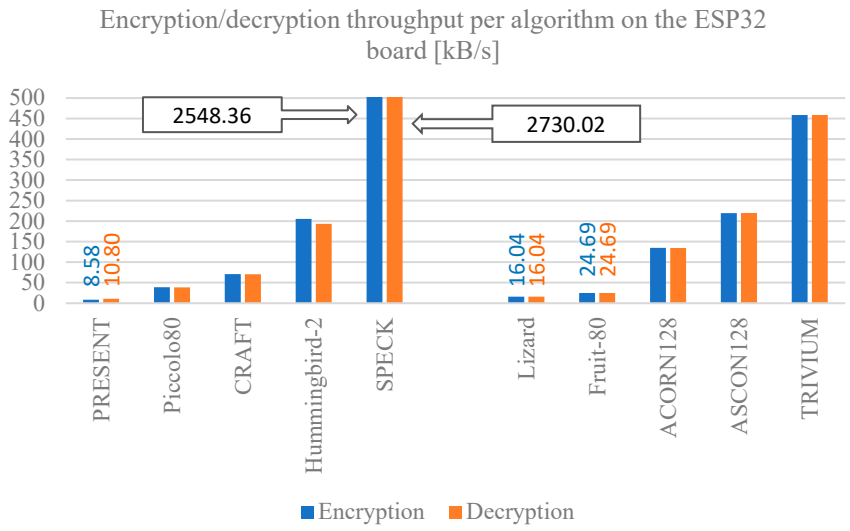


Figure 11. Encryption/decryption throughput per algorithm on the ESP32 board.

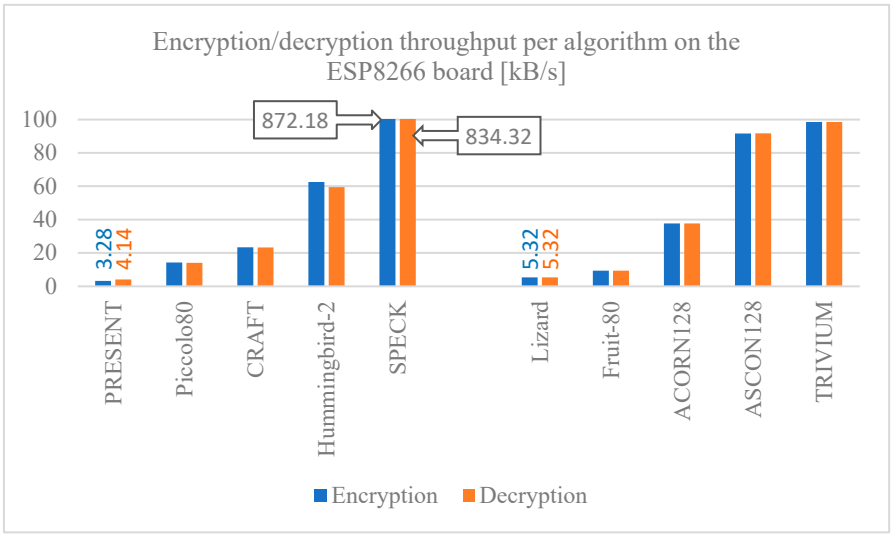


Figure 12. Encryption/decryption throughput per algorithm on the ESP8266 board.

The comparison among the results for the development boards show that, in general, the MSP430 Launchpad board has a lower throughput compared to the ESP32 and ESP8266 boards, so the latter offer better performance when implementing any of the algorithms shown if throughput is a major concern.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 October 2024 doi:10.20944/preprints202410.2545.v1

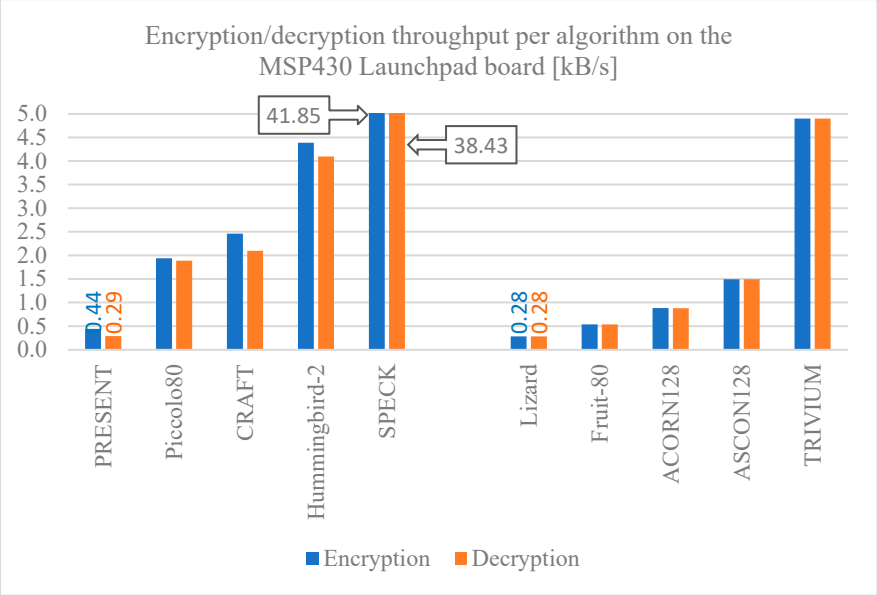


Figure 13. Encryption/decryption throughput per algorithm on the MSP430 Launchpad board.

5.4. Electric Current Usage

Figure 14a,b shows the average current usage of each development board during the execution of every algorithm for both encryption and decryption. The values obtained for each of the implemented algorithms are highly similar to each other. Therefore, no significant difference among algorithms is considered. However, if the current is compared among boards, it is evident that the MSP430 Launchpad board requires the least amount of current, while the ESP family boards use similar amounts of current. It is also noted that the electrical current requirement is consistent across the algorithms for encryption and decryption. For this reason, if the electrical current provided by a power supply is a concern parameter, the best alternative to implement any of the algorithms is the MPS430 Launchpad board.

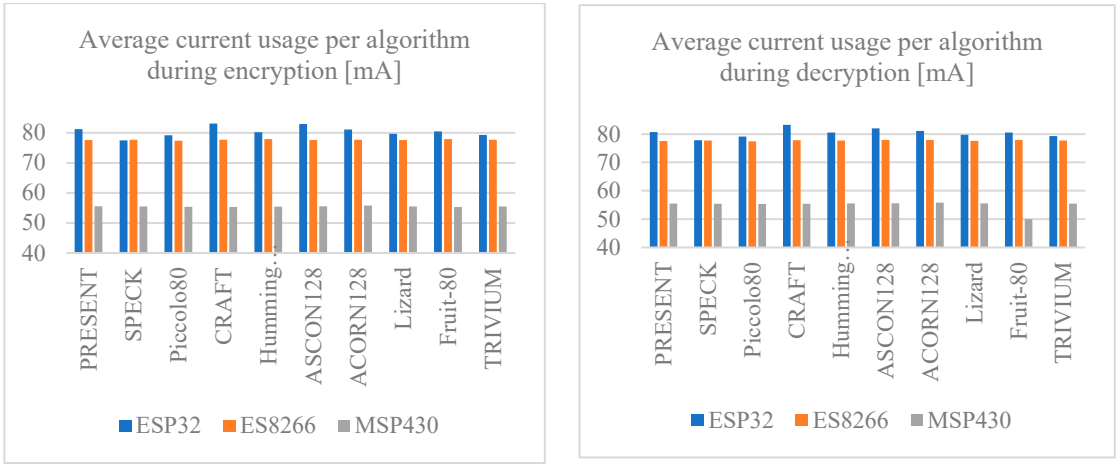


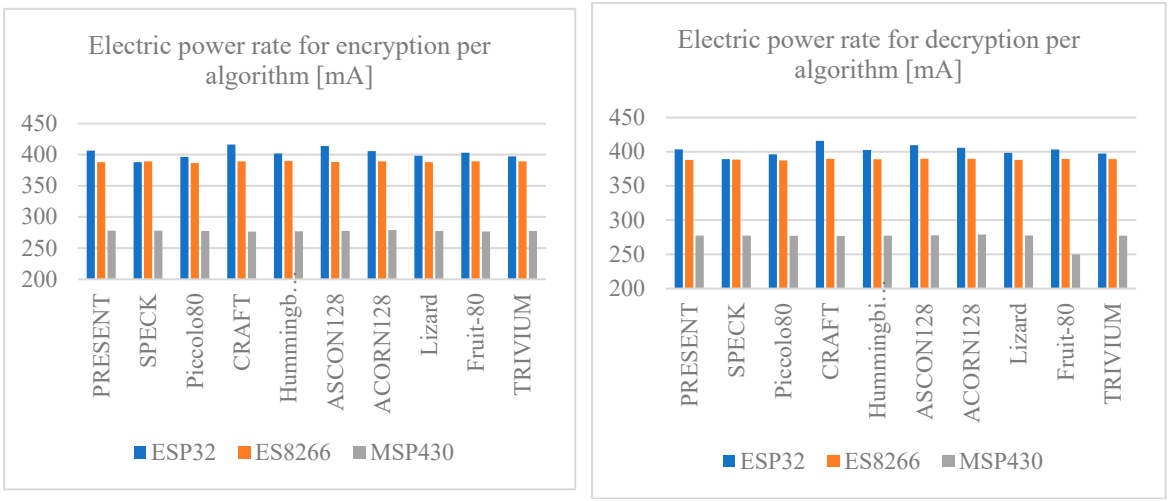
Figure 14. Average current usage per algorithm. **(a)** Encryption current usage; **(b)** Decryption current usage.

5.5. Electric Power Rate

Figure 15 shows the power requirement for all the development boards while running each of the algorithms for encryption and decryption. As in Figure 14, 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;

 doi:10.20944/preprints202410.2545.v1

the MSP430 Launchpad board requires less power compared to the ESP family boards, so it is a better option when power restrictions are present. The ESP family boards do not show significant differences in terms of power; however, the ESP32 board requires slightly more power than the ESP8266 board.



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

Figure 15. Electric power rate per algorithm. **(a)** Encryption power rate; **(b)** Decryption power rate.

5.6. Energy Usage Per Byte

Figure 16–18 provide the required energy to process a single byte of data, both encryption and decryption, for each cryptographic algorithm on the ESP32, ESP8266, and MSP430 Launchpad boards, respectively. The results reveal significant differences in energy required among the algorithms, as well as among the development boards used. The SPECK algorithm has a notably low energy consumption when implemented on any board, followed closely by the TRIVIUM algorithm, making both of them good options for applications where energy is a determining factor. On the contrary, the PRESENT algorithm has a much higher energy consumption per byte of data compared to the other algorithms, so it is not an ideal option for these cases.

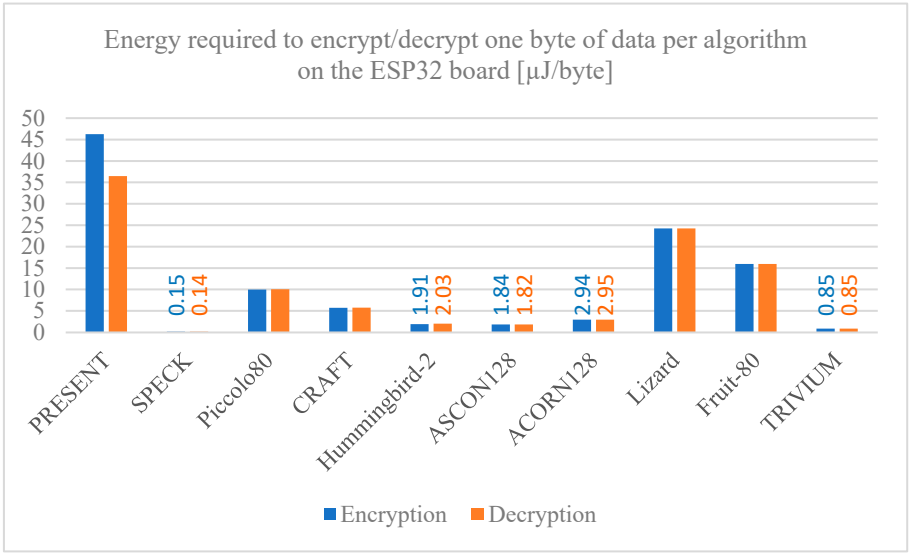


Figure 16. Energy required to encrypt/decrypt one byte of data per algorithm on the ESP32 board.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 October 2024 doi:10.20944/preprints202410.2545.v1

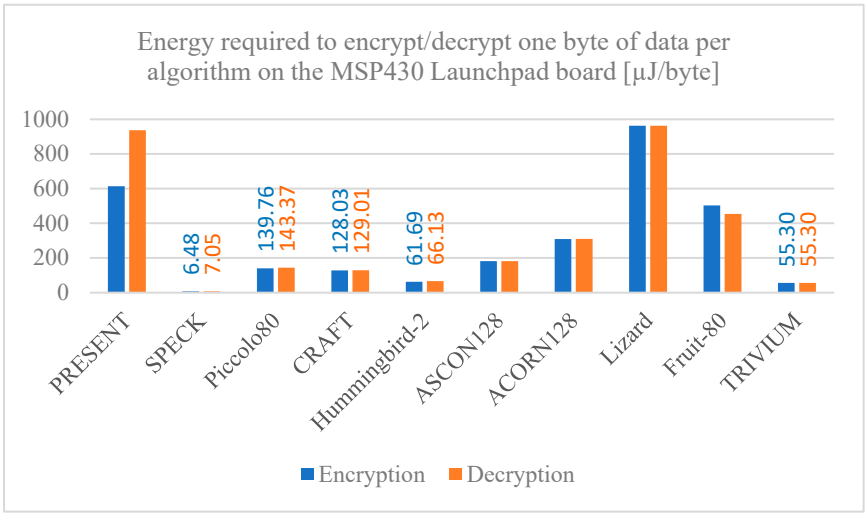


Figure 17. Energy required to encrypt/decrypt one byte of data per algorithm on the ESP8266 board.

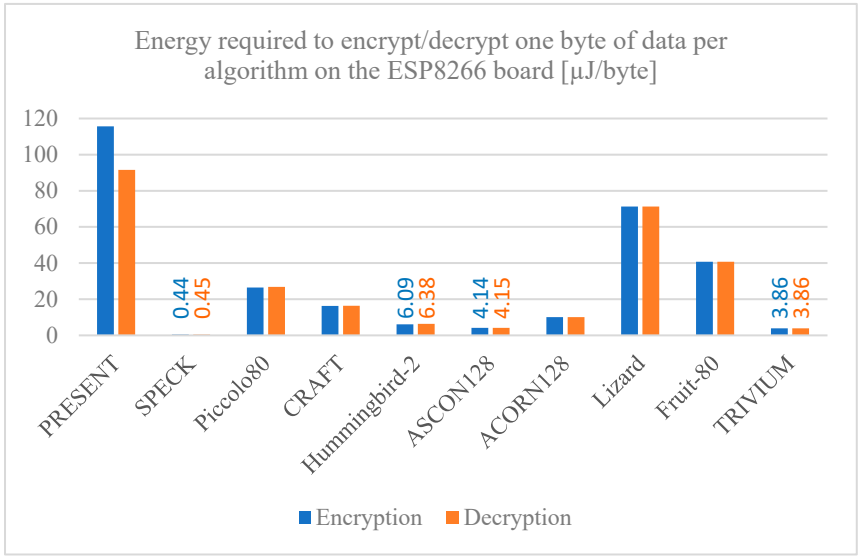


Figure 18. Energy required to encrypt/decrypt one byte of data per algorithm on the MSP430 Launchpad board.

Something worth mentioning is that for all the algorithms the MSP430 Launchpad board requires more energy per byte compared to the other boards evaluated. The reason for this is that, although the MSP430 Launchpad board has a lower power and current usage, it requires more time to encrypt and decrypt data, so it ends up using more energy to execute the algorithms compared to the ESP family boards.

6. Conclusions

Lightweight cryptography is an indispensable security mechanism for a secure implementation of the Internet of Things (IoT), given that it is of significant importance to guarantee the security and privacy of 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 since all the chosen algorithms were correctly implemented, and all the performance metrics defined were measured on the three selected development boards. 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. 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 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 evaluation of more cryptographic algorithms under development using more devices with different specifications, as well as considering 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. Lastly, while these mechanisms do help in protecting IoT systems against threats, it is crucial that security education and awareness are promoted among developers, users, and stakeholders in the IoT ecosystem, as all these mechanisms are useless if people are not aware of the risks that exist and how to identify them in order to avoid compromising security incidents.

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

Funding: PROFAPI 2024-0682.

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 to 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.

References

1. J. -Y. Yu, E. Lee, S. -R. Oh, Y. -D. Seo and Y. -G. Kim, "A Survey on Security Requirements for WSNs: Focusing on the Characteristics Related to Security," in IEEE Access, vol. 8, pp. 45304-45324, 2020. https://doi.org/10.1109/ACCESS.2020.2977778.
2. Tomić and J. A. McCann, "A Survey of Potential Security Issues in Existing Wireless Sensor Network Protocols," in IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1910-1923, Dec. 2017. https://doi.org/10.1109/JIOT.2017.2749883.
3. P. Sun, S. Shen, Y. Wan, Z. Wu, Z. Fang and X. -z. Gao, "A Survey of IoT Privacy Security: Architecture, Technology, Challenges, and Trends," in IEEE Internet of Things Journal. https://doi.org/10.1109/JIOT.2024.3372518.
4. M. Adil et al., "Survey: Self-Empowered Wireless Sensor Networks Security Taxonomy, Challenges, and Future Research Directions," in IEEE Sensors Journal, vol. 23, no. 18, pp. 20519-20535, 15 Sept.15, 2023. https://doi.org/10.1109/JSEN.2022.3216824.
5. F. Alawad and F. A. Kraemer, "Value of Information in Wireless Sensor Network Applications and the IoT: A Review," in IEEE Sensors Journal, vol. 22, no. 10, pp. 9228-9245, 15 May15, 2022. https://doi.org/10.1109/JSEN.2022.3165946.
6. E. Vieira, J. Almeida, J. Ferreira and P. C. Bartolomeu, "Enabling Seamless Data Security, Consensus, and Trading in Vehicular Networks," in IEEE Transactions on Intelligent Vehicles. https://doi.org/10.1109/TIV.2024.3388247.
7. B. Tahir, A. Jolfaei and M. Tariq, "A Novel Experience-Driven and Federated Intelligent Threat-Defense Framework in IoMT," in IEEE Journal of Biomedical and Health Informatics. https://doi.org/10.1109/JBHI.2023.3236072.

8. K. Islam, W. Shen and X. Wang, "Wireless Sensor Network Reliability and Security in Factory Automation: A Survey," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 6, pp. 1243-1256, Nov. 2012. https://doi.org/10.1109/TSMCC.2012.2205680.
9. H. A. D. Nguyen and Q. P. Ha, "Wireless Sensor Network Dependable Monitoring for Urban Air Quality," in IEEE Access, vol. 10, pp. 40051-40062, 2022. https://doi.org/10.1109/ACCESS.2022.3166904.
10. S. Surekha and M. Z. U. Rahman, "Cognitive Energy-Aware Spectrum Sensing With Improved Throughput for Medical Sensor Networks," in IEEE Sensors Letters, vol. 6, no. 6, pp. 1-4, June 2022, Art no. 5500904. https://doi.org/10.1109/LSENS.2022.3180629.
11. Z. Yu, H. Gao, X. Cong, N. Wu and H. H. Song, "A Survey on Cyber–Physical Systems Security," in IEEE Internet of Things Journal, vol. 10, no. 24, pp. 21670-21686, 15 Dec.15, 2023. https://doi.org/10.1109/JIOT.2023.3289625.
12. S. J. Saba, Bashar Talib Al-Nuaimi, and Ruaa Azzah Suhail, "A review of traditional, lightweight and ultra-lightweight cryptography techniques for IoT security environment," Jan. 2023. https://doi.org/10.1063/5.0103349.
13. Kamaldeep, M. Dutta and J. Granjal, "Towards a Secure Internet of Things: A Comprehensive Study of Second Line Defense Mechanisms," in IEEE Access, vol. 8, pp. 127272-127312, 2020. https://doi.org/10.1109/ACCESS.2020.3005643.
14. M. binti Mohamad Noor and W. H. Hassan, "Current research on Internet of Things (IoT) security: A survey," Computer Networks, vol. 148, pp. 283–294, Jan. 2019. https://doi.org/10.1016/j.comnet.2018.11.025.
15. G. Murtaza, F. Iqbal, A. Altaf and A. Rasheed, "Techniques for Resource-Efficient, Lightweight Cryptography in IoT Devices for Smart Environment," 2023 Sixth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU), Riyadh, Saudi Arabia, 2023, pp. 223-228. https://doi.org/10.1109/WiDS-PSU57071.2023.00053.
16. Kapalova, N., Algazy, K., & Haumen, A. (2023). Development of a new lightweight encryption algorithm. Eastern-European Journal of Enterprise Technologies, 3(9 (123), 6–19. https://doi.org/10.15587/1729-4061.2023.280055
17. Hughes, L.E. (2022). Basic Cryptography: Symmetric Key Encryption. In: Pro Active Directory Certificate Services. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7486-6_1
18. C. Paar and J. Pelzl, Understanding Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. https://doi.org/10.1007/978-3-642-04101-3.
19. K. Suwais, "Stream Cipher Based on Game Theory and DNA Coding," Intelligent Automation & Soft Computing, vol. 33, no. 3, pp. 1815–1834, 2022. https://doi.org/10.32604/iasc.2022.025076.
20. G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, and C. Manifavas, "A review of lightweight block ciphers," Journal of Cryptographic Engineering, vol. 8, no. 2, pp. 141–184, Apr. 2017. https://doi.org/10.1007/s13389-017-0160-y.
21. S. A. Jassim and A. K. Farhan, "A Survey on Stream Ciphers for Constrained Environments," 2021 1st Babylon International Conference on Information Technology and Science (BICITS), Babil, Iraq, 2021, pp. 228-233. https://doi.org/10.1109/BICITS51482.2021.9509883.
22. Thomas Xuan Meng and W. J. Buchanan, "Lightweight Cryptographic Algorithms on Resource-Constrained Devices," Sep. 2020. https://doi.org/10.20944/preprints202009.0302.v1.
23. Rameez Raja Kureshi and Bhupesh Kumar Mishra, "A Comparative Study of Data Encryption Techniques for Data Security in the IoT Device," Lecture notes in electrical engineering, pp. 451–460, Jan. 2022. https://doi.org/10.1007/978-981-16-7637-6_40.
24. Sevin and A. A. O. Mohammed, "A survey on software implementation of lightweight block ciphers for IoT devices," Journal of Ambient Intelligence and Humanized Computing, Jul. 2021. https://doi.org/10.1007/s12652-021-03395-3.
25. L. Jiao, Y. Hao, and D. Feng, "Stream cipher designs: a review," Science China Information Sciences, vol. 63, no. 3, Feb. 2020. https://doi.org/10.1007/s11432-018-9929-x.
26. V. A. Thakor, M. A. Razzaque and M. R. A. Khandaker, "Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities," in IEEE Access, vol. 9, pp. 28177-28193, 2021. https://doi.org/10.1109/ACCESS.2021.3052867.
27. O. G. Dorobantu, A. -G. Apostol and O. Datcu, "The poly-alphabetic substitution ciphers - a viable solution for IoT applications?," 2022 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 2022, pp. 1-4. https://doi.org/10.1109/ISETC56213.2022.10010173.
28. Bogdanov et al., "PRESENT: An Ultra-Lightweight Block Cipher," Cryptographic Hardware and Embedded Systems - CHES 2007, pp. 450–466. https://doi.org/10.1007/978-3-540-74735-2_31.
29. K. Jeong, Y. Lee, J. Sung, and S. Hong, "Improved differential fault analysis on PRESENT-80/128," International Journal of Computer Mathematics, vol. 90, no. 12, pp. 2553–2563, Dec. 2013. https://doi.org/10.1080/00207160.2012.760732.
30. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK Families of Lightweight Block Ciphers," ePrint IACR, 2013. https://eprint.iacr.org/2013/404.

31. F. Abed, E. List, S. Lucks, and J. Wenzel, “Cryptanalysis of the Speck Family of Block Ciphers,” Cryptology ePrint Archive (eprint.iacr.org), 2013. https://eprint.iacr.org/2013/568 (accessed Apr. 18, 2024).
32. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, “Piccolo: An Ultra-Lightweight Blockcipher.” Available: https://www.iacr.org/archive/ches2011/69170343/69170343.pdf.
33. Y. Liu, C. Liu, Z. Liu, W. Li, Q. Wang, and D. Gu, “Improved meet-in-the-middle attacks on reduced-round Piccolo,” Science China Information Sciences, vol. 61, no. 3, Nov. 2017. https://doi.org/10.1007/s11432-016-9157-y.
34. Y. Wang, W. Wu, and X. Yu, “Biclique Cryptanalysis of Reduced-Round Piccolo Block Cipher,” Lecture notes in computer science, pp. 337–352, Jan. 2012. https://doi.org/10.1007/978-3-642-29101-2_23.
35. Beierle, C., Leander, G., Moradi, A., & Rasoolzadeh, S. (2019). CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks. IACR Transactions on Symmetric Cryptology, 2019(1), 5–45. https://doi.org/10.13154/tosc.v2019.i1.5-45
36. K. Pang and S. F. Abdul-Latip, “Key-dependent side-channel cube attack on CRAFT,” ETRI Journal, vol. 43, no. 2, pp. 344–356, Mar. 2021. https://doi.org/10.4218/etrij.2019-0539.
37. D. Engels, M.-J. Saarinen, P. Schweitzer, and E. Smith, “The Hummingbird-2 Lightweight Authenticated Encryption Algorithm.” Accessed: Jun. 03, 2024. [Online]. Available: https://eprint.iacr.org/2011/126.pdf
38. M.-J. O. Saarinen, “Related-key Attacks Against Full Hummingbird-2,” Cryptology ePrint Archive (eprint.iacr.org), 2013. https://eprint.iacr.org/2013/070 (accessed Jun. 03, 2024).
39. C. D. Schläffer Maria Eichlseder, Florian Mendel, Martin, “Ascon – Authenticated Encryption and Hashing,” ascon.iaik.tugraz.at. https://ascon.iaik.tugraz.at/
40. T. L. Computer Security Division, “Announcing Lightweight Cryptography Selection | CSRC,” CSRC | NIST, Feb. 06, 2023. https://csrc.nist.gov/News/2023/lightweight-cryptography-nist-selects-ascon
41. L. Weissbart and S. Picek, “Lightweight but Not Easy: Side-channel Analysis of the Ascon Authenticated Cipher on a 32-bit Microcontroller,” Cryptology ePrint Archive (eprint.iacr.org), 2023. https://eprint.iacr.org/2023/1598 (accessed Apr. 18, 2024).
42. H. Wu, “ACORN: A Lightweight Authenticated Cipher (v3),” 2016. Available: https://competitions.cr.yp.to/round3/acornv3.pdf
43. X. Zhang and D. Lin, “Cryptanalysis of Acorn in Nonce-Reuse Setting,” Lecture notes in computer science, pp. 342–361, Jan. 2018. https://doi.org/10.1007/978-3-319-75160-3_21.
44. M. Hamann, M. Krause, and W. Meier, “LIZARD – A Lightweight Stream Cipher for Power-constrained Devices,” IACR Transactions on Symmetric Cryptology, pp. 45–79, Mar. 2017. https://doi.org/10.46586/tosc.v2017.i1.45-79.
45. Baksi, S. Kumar and S. Sarkar, "A New Approach for Side Channel Analysis on Stream Ciphers and Related Constructions," in IEEE Transactions on Computers, vol. 71, no. 10, pp. 2527-2537, 1 Oct. 2022. https://doi.org/10.1109/TC.2021.3135191.
46. “Fruit-80: A Secure Ultra-Lightweight Stream Cipher for Constrained Environments,” Entropy, vol. 20, no. 3, p. 180, Mar. 2018. https://doi.org/10.3390/e20030180.
47. Yosuke Todo, W. Meier, and K. Aoki, “On the Data Limitation of Small-State Stream Ciphers: Correlation Attacks on Fruit-80 and Plantlet,” Lecture notes in computer science, pp. 365–392, Jan. 2020. https://doi.org/10.1007/978-3-030-38471-5_15.
48. C. De Cannière and B. Preneel, “Trivium Specifications.” Available: https://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf.
49. Francisco Eugenio Potestad-Ordóñez, M. Valencia-Barrero, C. Baena-Oliva, P. Parra-Fernández, and Carlos Jesús Jiménez-Fernández, “Breaking Trivium Stream Cipher Implemented in ASIC Using Experimental Attacks and DFA,” Sensors (Basel), vol. 20, no. 23, pp. 6909–6909, Dec. 2020. https://doi.org/10.3390/s20236909.
50. Miguel Antonio Caraveo-Cacep, Rubén Vázquez-Medina, Antonio Hernández Zavala, A survey on low-cost development boards for applying cryptography in IoT systems, Internet of Things, Volume 22, 2023, 100743, ISSN 2542-6605. https://doi.org/10.1016/j.iot.2023.100743.
51. P. Plaza, Elio Sancristobal, G. Carro, M. Castro, and Elena Ruiz Ruiz, “Wireless Development Boards to Connect the World,” Lecture notes in networks and systems, pp. 19–27, Jan. 2018. https://doi.org/10.1007/978-3-319-64352-6_2.
52. Singh, Dhawan & Sandhu, Amanpreet & Thakur, Aditi & Priyank, Nikhil. (2020). An Overview of IoT Hardware Development Platforms.
53. “Arduino Cryptography Library: Arduino Cryptography Library,” rweather.github.io. https://rweather.github.io/arduinolibs/crypto.html
54. P. Tonkovic, “Pepton21/present-cipher,” GitHub, Oct. 15, 2023. https://github.com/Pepton21/present-cipher (accessed Apr. 19, 2024).

 doi:10.20944/preprints202410.2545.v1

55. P. Jovanovic, "Daeinar/piccolo," GitHub, Mar. 02, 2022. https://github.com/Daeinar/piccolo (accessed Apr. 19, 2024).
56. P. Jovanovic, "Daeinar/piccolo," GitHub, Mar. 02, 2022. https://github.com/Daeinar/piccolo (accessed Apr. 19, 2024).