


A Comparative Evaluation of Symmetric Cryptography Algorithms for Resource-Constrained Devices

Mayksuel Ramalho   [Universidade Federal Fluminense | mayksuelramalho@id.uff.br]

Gabriel Sampaio  [Universidade Federal Fluminense | gabrielsampaio@id.uff.br]


Nicholas Neves  [Universidade Federal Fluminense | nicholasneves@id.uff.br]

Rafael Porto  [Universidade Federal Fluminense | rafaelporto@id.uff.br]

Victor Afonso Sobral  [Universidade Federal Fluminense | victorafonso@id.uff.br]

Marcos Rezende  [Prefeitura Municipal de Niterói - Defesa Civil | marcosrfd.01@gmail.com]

Dianne S. V. Medeiros   [Universidade Federal Fluminense | diannescherly@id.uff.br]

 School of Engineering, Universidade Federal Fluminense, R. Passo da Pátria, 156, São Domingos, Niterói, RJ, 24210-240, Brazil.

Received: 04 September 2024 • **Accepted:** 11 March 2025 • **Published:** 13 May 2025

Abstract Data security in the Internet of Things (IoT) is crucial for protecting both the devices and the data they transmit over the network. Nevertheless, security is often overlooked in this context, leaving systems vulnerable to cyberattacks that can compromise information confidentiality and integrity. This work focuses on a use case of an environment remote monitoring system for disaster prevention, in which information must be confidential and intact. The AES (Advanced Encryption Standard) and Speck cryptographic algorithm families are evaluated in both traditional and memory-optimized implementations, targeting data confidentiality. The algorithms are assessed through practical experiments on two resource-constrained hardware platforms. Transmission throughput, estimated energy consumption, execution time, and memory usage are evaluated. Results show that the Speck family executes more quickly, has lower estimated energy consumption, and occupies less memory space than AES in both platforms.

Keywords: Arduino, Remote monitoring, Internet of Things, Symmetric cryptography

1 Introduction

The Internet of Things (IoT) represents a revolution in how people interact with the world (Mouha and Ait [2021]). IoT offers an infrastructure for real-time remote data collection and process automation (Hossein Motlagh *et al.* [2020]). Nevertheless, the increasing use of IoT devices in remote monitoring systems exposes these systems to malicious agents. Such agents can explore the vulnerabilities to harm the system. Many of these systems focus on collecting and transmitting monitored data that should be protected. Hence, it is necessary to address vulnerabilities caused by the use of weak security protocols and policies (Tawalbeh *et al.* [2020]). Data protection is important even when the monitored data is not sensitive, such as in environmental monitoring, as vulnerabilities can undermine data reliability.

In natural disaster scenarios, for example, the lack of robust early warning systems for extreme weather events is a global concern. Inadequate monitoring can result in significant property damage, loss of human lives, and devastating socioeconomic impacts. Therefore, it is imperative to develop effective and secure solutions to mitigate the risks associated with such disasters (Alvalá and Barbieri [2017]). The accuracy and integrity of data obtained from these systems are crucial to underpin an efficient decision-making process aimed at ensuring people's safety in risky areas. The lack of robust security can jeopardize the effectiveness of monitoring systems and the safety and well-being of affected people. Thus, the vulnerability of remote monitoring systems should

be mitigated from the beginning of the system's design.

The development of secure IoT systems is challenging, primarily due to the devices' energy and computational resource constraints. These constraints hinder the adoption of secure protocols that involve robust encryption mechanisms. Despite this difficulty, it is essential to integrate cryptographic techniques into IoT systems. The goal is to ensure the confidentiality and integrity of the data transmitted in remote monitoring systems. However, cryptographic techniques add overhead that can impair system performance, either by requiring computational capacity beyond the one available, increasing energy consumption and thereby reducing the system's average lifespan, or by increasing the total communication latency. In this scenario, it is essential to seek lightweight cryptographic algorithms to incorporate into IoT systems. This work focuses on a use case of remote monitoring for natural disaster prevention, in which IoT devices are strategically distributed to collect and transmit environmental data in real-time and securely. In this context, the confidentiality and integrity of collected and stored data are fundamental to composing a historical series that allows the creation of effective public policies. Hence, this work evaluates the impact of using cryptographic algorithms on the communication performance and lifespan of remote monitoring systems.

Several works in the literature conduct a comprehensive evaluation comparing various cryptographic techniques using multiple platforms (Pereira *et al.* [2017]; Fotovvat *et al.* [2021]) or simulation (Beg *et al.* [2019]). Other works fo-

cus on evaluating lightweight cryptographic algorithms implemented on Arduino UNO and Raspberry Pi 3 hardware platforms (El-hajj *et al.* [2023]). This work claims that a security layer must be added to data transmission in a remote monitoring system for natural disaster prevention to ensure confidentiality. As such, we incorporate traditional and lightweight encryption techniques on resource-constrained devices, considering the original and optimized implementations of two well-known encryption algorithms. The novelty of this work lies in the comparative evaluation between the lightweight encryption algorithm, Speck, and the classic algorithm, AES (Advanced Encryption Standard), in their traditional and optimized versions, and varying the encryption key size. The optimized versions of the AES and Speck algorithms aim to consume less memory on the device and are referred to as “Tiny” and “Small”. The comparison is made through practical experiments using popular IoT devices, the Arduino UNO and WeMos D1 platforms. The influence of each algorithm and the encryption key size on the system’s performance is assessed in terms of encryption time, data throughput, energy consumption, SRAM (Static Random-Access Memory) occupancy, and flash memory occupancy. The results show no variation in transmission throughput due to the use of symmetric encryption for both IoT platforms. The estimated energy consumption for the non-optimized Speck algorithm is lower than the others independently of the platform. The percentage of memory occupancy is low, although higher than that of the optimized versions. Moreover, the WeMos D1 is more efficient in terms of energy consumption compared to the Arduino UNO.

The contributions of this work are three-fold, all focused on performance comparison:

- We evaluate classic (AES) and lightweight (Speck) algorithms to encrypt and decrypt data using traditional and optimized implementations;
- We verify the influence of key size on the performance of all algorithms;
- We investigate how the hardware limitation affects the performance of each algorithm by testing two hardware platforms, one of them being much more resource-constrained compared to the other.

The remainder of this work is organized as follows. Section 2 discusses related work. Section 3 introduces the cryptographic algorithms used. Section 4 presents the evaluation methodology. Section 5 discusses the results. Finally, Section 6 concludes this work and presents directions for future research.

2 Related Work

The security of IoT systems is an important topic given the rapid expansion of applications related to the IoT paradigm. However, applying cryptographic techniques to these systems is challenging due to the computational and energy constraints of the devices that are part of the system. Several studies evaluate the performance of cryptographic algorithms, aiming to create a benchmark. The idea is to facilitate

the selection of the most suitable algorithms for different systems.

Guinelli *et al.* [2018] conduct a comparative analysis between the Rijndael and Serpent encryption algorithms, considering the following evaluation metrics: storage usage, clock cycles, energy consumption, memory usage, and execution time. The idea is to determine in which scenario and for which requirement each algorithm is most suitable. The algorithms are implemented on an Arduino Mega 2560 platform, and the results show that the Rijndael algorithm offers significant advantages in terms of computational efficiency, consuming fewer clock cycles and processing time compared to Serpent. Nevertheless, Serpent consumes less SRAM memory and exhibits lower energy consumption during algorithm execution, suggesting that it is more suitable for applications with memory and energy constraints. Differently, this work also focuses on lightweight ciphers, such as the Speck algorithm.

Zanon *et al.* [2022] address the vulnerability of wearable heart monitoring devices, especially those integrated into the Internet of Medical Things (IoMT). The authors focus on a scenario of secure data transmission by wearable devices during heart exams. They also highlight the need to ensure the security of data transmitted by wearable devices during such exams due to data sensitivity and value. The authors implement a security layer based on lightweight cryptography and evaluate the performance of different encryption algorithms in this context. The algorithms evaluated are AES-256 CBC (Cipher Block Chaining) implemented in hardware on the ESP32 microcontroller and two lightweight cryptography algorithms based on software, Speck and CLEFIA (Cipher for Lightweight Encryption Fast with Integer Approximation). The authors consider the performance and ease of implementation for each algorithm. Performance is evaluated in terms of throughput, latency, and energy consumption. The results show that AES-256 CBC performs well in terms of security and energy efficiency, but with relatively longer execution times. Speck and CLEFIA, as lightweight algorithms, offer lower energy consumption and execution time. In this work, we also compare AES and Speck, but we include the optimized implementations of these algorithms. Additionally, we expand the performance evaluation metrics to comprise memory usage.

Albarelo *et al.* [2020] evaluate the performance of three different algorithms that have distinct purposes: AES, SHA, and x25519, on IoT devices with limited resources. The authors propose a protocol that uses the x25519 algorithm to securely exchange keys between IoT devices in a network. This protocol enables secure communication between devices without the need to pre-insert keys into the devices, making it efficient in terms of execution time and security. The results show that the Arduino Uno R3 satisfactorily executed the AES and SHA algorithms, even with low processing power. The proposed key exchange protocol based on the x25519 algorithm also proved to be viable and effective, making it suitable for secure communication between IoT devices and a gateway. Like Guinelli *et al.*, the authors do not evaluate lightweight encryption algorithms such as Speck. Conversely, besides evaluating Speck, we also explore the memory-optimized versions of Speck and AES and investi-

gate the impact of key size on the algorithms' overall performance.

Vaz *et al.* [2023] propose an optimization for the AES algorithm to adapt it for execution on devices with constrained computational resources. The authors modify two functions of the algorithm to use a smaller substitution matrix than the original algorithm and less costly operations to generate the state matrix. The optimizations focus mainly on the Sub-Bytes operation, which replaces each byte of the plain text block with another byte using a predefined substitution table, and MixColumns, which combines each column of the cipher text block using a fixed coefficient matrix to mix the bytes. The idea is to improve execution time and memory consumption. The authors compare the proposal with the original algorithm using an ARM Cortex-M0+ microcontroller based on the SAM D21 chip from Microchip Technology. The authors find that the proposal reduces the average execution time by 86.71%, program memory consumption by 31.82%, and dynamic memory consumption by 89.04%. The authors do not compare the proposal with other algorithms optimized for use in resource-constrained devices. Distinctly, we compare optimized versions of AES and Speck and also the traditional implementation of Speck in terms of throughput and energy consumption. We also analyze the influence of key size on the performance of each algorithm.

Fotovat *et al.* [2021] comprehensively evaluate the performance of 32 lightweight cryptographic algorithms on IoT platforms such as Raspberry Pi 3, Raspberry Pi Zero W, and iMX233. The algorithms are evaluated in terms of memory usage, energy consumption, and execution time. The results show that execution time and energy consumption vary significantly between algorithms, but memory and processing consumption remain similar. Despite the wide variety of algorithms tested, the Speck and AES algorithms are not evaluated. Conversely, we evaluate these algorithms and their memory-optimized versions.

Pereira *et al.* [2017] evaluate reference implementations of various symmetric encryption algorithms, cryptographic hash functions, message authentication codes, and authenticated encryption with associated data. The algorithms are evaluated on two IoT platforms, TelosB and Intel Edison, and three popular IoT operating systems, Yocto, ContikiOS, and TinyOS. The comparative evaluation is performed in terms of execution time and energy consumption. Among the symmetric encryption algorithms evaluated, AES has the shortest execution time and lowest energy consumption on the TelosB platform running the TinyOS or ContikiOS operating systems. For the Intel Edison platform running the Yocto operating system, AES encryption has the shortest execution time and memory consumption, but only if the messages are smaller than 40 bytes. Similarly, we also focus on resource-constrained devices. However, we enrich the available benchmark by evaluating the lightweight cipher Speck, and the memory-optimized versions of both AES and Speck, also varying their key size.

El-hajj *et al.* [2023] evaluate the performance of symmetric encryption algorithms on the Arduino UNO and Raspberry Pi 3 platforms in terms of speed, cost, and energy efficiency. The algorithms evaluated include AES and Speck. The results show that the Speck family algorithms are among

the best performers in terms of energy consumption, speed, and memory usage. However, the AES implementation evaluated is not optimized for reduced memory consumption, which is different from our work.

Panahi *et al.* [2021] evaluate the performance of lightweight cryptographic algorithms and AES. The evaluation is conducted using the Arduino MEGA 2560 and Raspberry Pi 3 platforms. The idea is to compare memory usage, energy efficiency, throughput, and execution time of the algorithms. The results show that AES performs well, but it is not the algorithm with the lowest energy consumption and execution time. Additionally, on the Raspberry Pi 3 platform, it is the algorithm with the highest RAM usage, and on the Arduino Mega platform, it has the highest ROM (Read-Only Memory) consumption. Although the authors conduct an extensive investigation of lightweight cryptographic algorithms and provide a comprehensive benchmark, their analysis does not include Speck. In contrast, this work evaluates not only Speck but also memory-optimized versions of AES and Speck, broadening the scope of analysis to include additional algorithms tailored for resource-constrained environments.

Munoz *et al.* [2018] analyze the cipher duration and energy consumption of the AES algorithm, implemented in both software and hardware with various key and buffer size settings on the CYW and BCM IoT boards. The CYW board is best suited for computation-heavy workloads and provides a hardware accelerator for cryptographic operations. In turn, the BCM board is more adequate for low-cost and scalable applications. The results show that hardware implementation is more sensitive to buffer size, consuming less time and energy only when the buffer size is large enough. Increased key size leads to higher resource consumption, and the CYW board performs better than the BCM board. In contrast, besides classic AES, we evaluate a lightweight encryption algorithm, Speck, and optimized versions of AES and Speck, focusing on reducing memory consumption. Our work uses IoT boards that are more limited in terms of computational and energy resources, particularly the Arduino UNO R3. Unlike the CYW, both IoT boards evaluated in our work lack hardware acceleration for cryptography. Additionally, we explore other performance metrics, such as throughput and memory consumption, providing a broader understanding of the practical applicability of cryptographic algorithms in highly resource-constrained environments.

Sleem and Couturier [2020] propose Speck-R, an ultra-lightweight cryptographic algorithm based on Speck. The proposal is a hybrid cipher that combines the ARX (Addition/Rotation/XOR) architecture with a key-dynamic substitution layer. The authors compare Speck-R to Speck in terms of security and performance, focusing on encryption time across three IoT boards, including a highly resource-constrained device. The results show that compared to Speck, the execution time of Speck-R is reduced by 18% to 77% on resource-constrained devices while maintaining high security. In our work, we also target resource-constrained devices but expand the analysis by evaluating additional metrics, including decryption time, throughput, energy consumption, and memory usage. We further compare classic Speck's performance to the classic and optimized AES, offering a

deeper insight into their practical suitability for resource-constrained environments.

Maitra *et al.* [2019] evaluate the performance of AES and XTEA (eXtended Tiny Encryption Algorithm) on resource-constrained devices, comparing also hardware-accelerated and software-implemented AES. The authors focus their analysis on memory usage, code size, energy consumption, and execution time. The results indicate that while hardware-accelerated AES is the fastest and most energy-efficient, XTEA shows comparable execution time and energy consumption. Compared to software-implemented AES, XTEA achieves better performance with the added benefit of using much less memory. Therefore, the authors conclude that XTEA is a suitable option for platforms with limited memory or without AES-enabled hardware acceleration. Similarly, our work investigates resource-constrained devices and compares a classic cipher with a lightweight one. However, unlike Maitra *et al.*'s work, the devices used in this paper lack cryptographic hardware acceleration. Additionally, we expand the evaluation by assessing throughput performance, a memory-optimized AES version, and the Speck lightweight cipher, along with its optimized version. We also assess how key size impacts the performance of each algorithm, offering valuable insights for application developers to select the most suitable encryption method.

Qasaimieh *et al.* [2021] propose LAES (Lightweight Advanced Encryption Standard), a lightweight variant of the AES algorithm. The proposal is evaluated in terms of randomness, CPU cycles, consumed power, and the amount of charge required. The authors compare LAES with other cryptographic algorithms implemented on an Arduino UNO board. The results indicate that LAES outperforms AES, Present, and CLEFIA in terms of CPU cycles, power consumption, and charge usage for block encryption while maintaining competitive randomness levels and processing times. In the same direction, we also evaluate lightweight and classic cryptographic algorithms in a resource-constrained context. Conversely, our work expands upon Qasaimieh *et al.*'s by including additional performance metrics, such as throughput and memory usage. We also evaluate the Speck, its optimized version, and AES's optimized version, which the authors do not analyze. Lastly, we vary the algorithms' key size and evaluate its impact on each performance metric.

Unlike the works cited, this work compares the AES and Speck algorithms in their traditional and optimized versions, varying the encryption key size. The evaluation is carried out through practical experiments on two hardware platforms, a highly resource-constrained device, the Arduino UNO R3, and the WeMos D1 ESP8266, which is limited in terms of processing and storage resources but offers greater capacity compared to the Arduino UNO R3.

3 Symmetric Cryptography Algorithms: AES and Speck

The cryptographic algorithms evaluated in this work belong to two families of symmetric encryption algorithms, AES and Speck. Symmetric encryption uses a single key that serves both to encrypt and decrypt data, being less complex

than asymmetric encryption. AES is known for its robustness and security, with different key sizes that allow adjusting the level of security to the availability of computational resources. AES operates by separating plain text blocks that are subjected to alternating rounds of substitutions and permutations. The block sizes can vary between 128, 192, and 256 bits, which represent the key sizes. In each encryption round, the functions of byte substitution (SubBytes), row shifting (ShiftRows), column mixing (MixColumns), and round key addition (AddRoundKey) are executed. For decryption, the inverse operations are performed, except for the AddRoundKey operation, with the first operation being ShiftRows instead of SubBytes. The last rounds, both in encryption and decryption, do not involve the MixColumns operation. The operations executed involve manipulating a byte matrix that is modified in each round. The number of rounds varies depending on the key size. For example, for 128-bit keys, 10 rounds are performed, while for 256-bit keys, 14 rounds are performed.

The SubBytes, ShiftRows, and MixColumns operations transform the matrix according to nonlinear and linear functions. The AddRoundKey step performs an XOR operation between the matrix and the round key (Abdullah [2017]). There are optimized variants of AES for devices with limited resources. Some of these variants have reference implementations in publicly available libraries. In this work, two variants are used to optimize memory consumption, namely "Tiny" and "Small," which aim to improve performance and reduce the device's resource usage. The "Tiny" optimization aims to minimize code size or the amount of memory needed, simplifying the algorithm or using compression techniques. On the other hand, the "Small" optimization seeks to balance efficiency and resource usage, maintaining an acceptable compromise between performance and the amount of memory required.

The Speck algorithm family was specifically designed for resource-constrained devices, providing security and efficiency in resource consumption. Speck is a lightweight block cipher that operates on fixed-size data blocks, usually of 64 or 128 bits. It consists of a series of substitution and permutation iterations, where the data is repeatedly transformed using the secret key. The algorithm uses a modified Feistel structure, which divides the input block into two halves and applies substitution and permutation operations on each half alternately, combining them at the end of each iteration. Speck is highly optimized for hardware implementation and offers a good balance between security and efficiency. Even though the Speck family is already focused on resource-constrained devices, there are also "Small" and "Tiny" variants available in public libraries, with the same objectives mentioned for the AES variants (de Paiva *et al.* [2023]).

AES encryption has already undergone extensive scrutiny by cryptanalysts. The Speck encryption is more recent and still needs to be extensively tested. Considering the 256-bit key, the AES security margin is null, as the 14 necessary rounds were broken in 2011 (Bogdanov *et al.* [2011]). In the Speck family, the most effective attack revealed a security margin of 26% (Song *et al.* [2016]).

4 Methodology

This work is part of a project funded by the Niterói City Hall, aimed at developing a remote monitoring system for natural disaster prevention capable of securely transmitting environmental monitoring data. Resource-limited devices are used to build remote sensors that capture environmental data, encrypt this data, and transmit it to a data storage and processing server via an IoT gateway.

The remote sensors in the project use the Arduino UNO R3 platform as hardware for simplicity and cost, but other hardware, such as the WeMos D1 ESP8266, could also be used. The hardware platform is integrated with an RTC module model DS 3231, a rain gauge model DAVIS 6464M, and a LoRa (Long Range) E32TTL100D module for low-power, long-range communication. The confidentiality of the transmitted data is ensured using a symmetric encryption algorithm. The IoT gateway consists of a LoRa E32TTL100D module and an RTC model DS 3231 module integrated with an Arduino UNO R3, which connects to a Raspberry Pi 4 Model B via a serial port. The Arduino UNO R3, which is part of the IoT gateway, is responsible for receiving the data, decrypting it, and sending it to the Raspberry Pi 4, which, in turn, is responsible for communication with the server through the MQTTS (Message Queuing Telemetry Transport Secured) protocol. Figure 1 shows the communication scheme between the remote sensor and the server via the IoT gateway, considering the use of Arduino UNO for the remote sensor. This work focuses on evaluating the encryption algorithms used for communication between the remote sensor and the IoT gateway, as highlighted by the orange-shadowed area in the figure.

The Speck and AES algorithms are evaluated with different key sizes and optimized versions, on both WeMos D1 and Arduino UNO. The existing reference implementations from the `crypto` library¹, developed for the Arduino platform, are used. The AES family includes the traditional implementation, AES, and the optimized versions, AES Small and AES Tiny. Similarly, the Speck family has the traditional version, Speck, and the optimized versions, Speck Small and Speck Tiny. Keys of 128, 192, and 256 bits are used whenever possible. It is worth noting that the “Tiny” variants do not implement a decryption function.

The encrypted data represents a packet to be transmitted using the LoRa module. This packet contains four different types of variables representing the environmental data collected by the remote sensor. Each variable is 4 B, totaling 16 B of data to be sent. The data is transmitted every 5 s, and the communication throughput, average execution time, estimated energy consumption, flash memory usage, and SRAM usage are evaluated. The flash memory stores the code, imported library files, literal constants, and strings. The SRAM temporarily stores the running program code, variables, functions, and the execution stack of the code. Thus, SRAM is the dynamic memory used for code execution. The WeMos D1 ESP8266 can operate in two CPU frequencies, 80 and 160 MHz. We evaluate both modes, as the increase in the CPU frequency could lead to higher energy consumption.

In an idle state, the Arduino UNO R3, along with all devices comprising the remote sensor, uses about 6 μA of current, considering an observation time of 10 seconds. In turn, the idle WeMos D1 ESP8266 consumes 4 μA of current when operating at either 80 or 160 MHz, considering the same observation time. The current is measured using a Minipa ET-1400 multimeter. The energy consumption can be estimated by $v \times i \times \Delta t$, where v is the supplied voltage, i is the consumed current, and Δt is the observation time. Considering $v = 12 V$, $i_{UNO} = 6 \mu A$, and $\Delta t = 10/3600 h$, the estimated energy consumption for the idle Arduino UNO R3 is 0.20 μWh . The same values for v and Δt can be used for the WeMos D1 ESP8266. Consuming a current of $i_{ESP80MHz} = i_{ESP160MHz} = 4 \mu A$, the estimated energy consumption for the idle WeMos D1 ESP8266 is 0.13 μWh . The peak consumption occurs when the data is transmitted by the communication module, reaching for the Arduino UNO R3 an average of $i_{UNO} = 14 \mu A$ during $\Delta t = 0.480/3600 h$, and therefore, an estimated consumption of 0.02 μWh . For the WeMos D1 ESP8266 the average current is $i_{ESP80MHz} = i_{ESP160MHz} = 6 \mu A$ during $\Delta t = 0.480/3600 h$, and therefore, an estimated consumption of 0.0096 μWh .

5 Results and Discussion

The communication between the remote sensor and the IoT gateway is evaluated to determine the best cryptographic algorithm for the remote monitoring scenario studied. It is important to note that the LoRa communication module used implements only the physical layer, unlike the LoRaWAN modules. While the LoRaWAN protocol uses AES encryption for data transmission, there is no default data encryption at the LoRa physical layer. In this work, the data to be transmitted is transformed into a stream of pre-encrypted bytes to be sent by the LoRa module. It is important to highlight that in all the results obtained when evaluating the receiver, the Tiny versions of the algorithms are omitted because there is no implementation of the decryption function, making it impossible to evaluate the investigated metrics.

In the first experiment, the throughput is evaluated. The throughput represents the data transfer rate between the transmitter and receiver devices. This metric is crucial for assessing each algorithm’s ability to handle a continuous flow of information. In this work, throughput is measured in packets per second. The throughput achieved for data transmission is indifferent to the encryption used, being equal to 3.4 packets per second. This occurs because the time required to execute the cryptographic algorithms is much shorter than the interval between packet transmissions, not affecting the total number of packets sent.

The execution time of the algorithms directly impacts the system’s latency and the device’s energy consumption. Figure 2 shows the algorithms’ execution times. Considering the Arduino UNO R3, the Speck 128 algorithm takes an average of 162.4 μs to encrypt or decrypt the data. When using the 192-bit and 256-bit keys, an increase of 5.6 μs and 9.6 μs is observed, respectively. Considering the WeMos D1 ESP8266, the Speck algorithm with key sizes of 128, 192,

¹Available at <https://rweather.github.io/arduino-libraries/crypto.html>

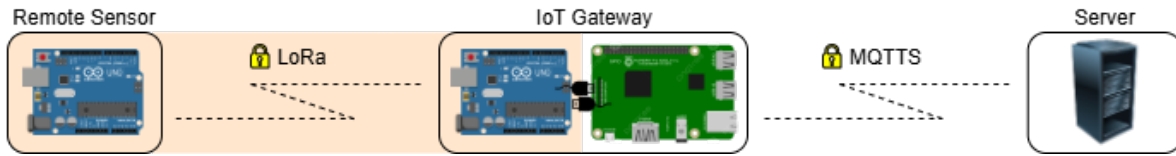


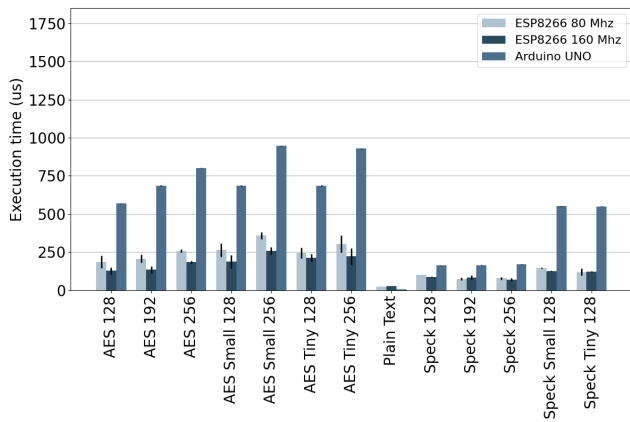
Figure 1. Communication scheme. The sensor and the gateway communicate via a LoRa module, with an additional security layer for encryption. The orange-shadowed area highlights the communication focus of this work.

and 256 bits takes less than $120\mu s$ to encrypt or decrypt the data. The operating CPU frequency has a small influence on the execution time. At 160 MHz, encryption and decryption are slightly faster. The unoptimized Speck algorithms are much faster than the Small and Tiny variants in both platforms. This shows that, despite aiming to reduce memory usage, optimized implementations increase execution time. The traditional AES algorithm also shows lower execution times than its Small and Tiny versions for corresponding key sizes. It is also observed that the larger the key used for AES, the longer the algorithm takes to execute. In decryption, most confidence intervals are not shown. This happens because there was no variation in execution time in the five rounds carried out. It is interesting to note that decryption takes longer than encryption for the AES family algorithms, while for the Speck family, the time required for both tasks is very similar. For AES 128, the encryption time is $569\mu s$ in Arduino UNO, increasing to $686\mu s$ with AES 192 and $802\mu s$ with AES 256. It is noteworthy that for the WeMos

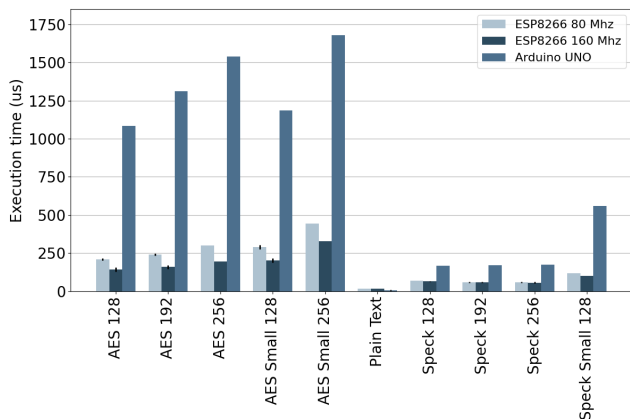
D1 ESP8266 the difference in execution time for the AES family considering both operating frequencies is much more significant than for the Speck family. This is probably due to the higher complexity of AES. We also observe that the worst case for the traditional AES occurs for a key size of 256 bits. In this case, the algorithm takes around $250\mu s$ to execute. For the AES' optimized versions, the worst case is found for AES Small 256, achieving almost $375\mu s$ of execution time. Despite the significant variation in the execution times of the algorithms, the influence of this time on data throughput using LoRa is negligible. Thus, any algorithm would be suitable. If it is necessary to minimize execution time, the Speck algorithm with a 128-bit key is the most appropriate for the Arduino UNO, but the WeMos D1 ESP8266 can be used with any key size.

The time spent initializing the device, preparing it for transmission or reception, and disregarding cryptographic operations is also evaluated. This assessment is important to verify the influence of cryptographic algorithms on system latency. The evaluation reveals that the initialization time is approximately $104ms$ for the Arduino UNO transmitter, while the Arduino UNO receiver requires approximately $100ms$. For the WeMos D1, the initialization times are $120ms$ for the transmitter and $422ms$ for the receiver. The initialization time and the execution time of the cryptographic algorithms directly influence the total communication latency. However, compared to the initialization time, the execution time of the algorithms is negligible. This evaluation corroborates the result obtained for the throughput, which is constant regardless of the algorithm used.

We evaluate the energy consumed by the system during the data transmission and reception process. With IoT devices often operating with limited energy resources, it is essential to choose algorithms that minimize energy consumption, ensuring the system's operational efficiency. The energy consumption estimate is based on current observations during encryption time in the transmitter and decryption time in the receiver. The experiment is repeated five times, and the observation time is extremely short, in the order of microseconds. There is often no variation in current during the observed time. For this reason, some results do not present confidence intervals. Energy consumption is extremely low for all algorithms, being slightly lower for decryption. The lowest estimate obtained for encryption using Arduino UNO, as shown in Figure 3(a), occurs for the Speck 128 algorithm. The use of 192-bit and 256-bit keys in this algorithm slightly increases the energy consumption estimate. For the AES family, the lowest estimate is obtained for the 128-bit key. The WeMos D1 ESP8266 is more efficient in terms of energy consumption. Encryption in this microcontroller takes less than $7pWh$, even for AES. The Speck family consumes



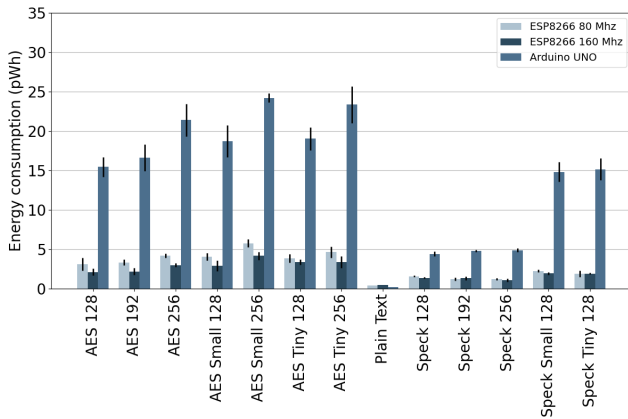
(a) Transmitter, encryption.



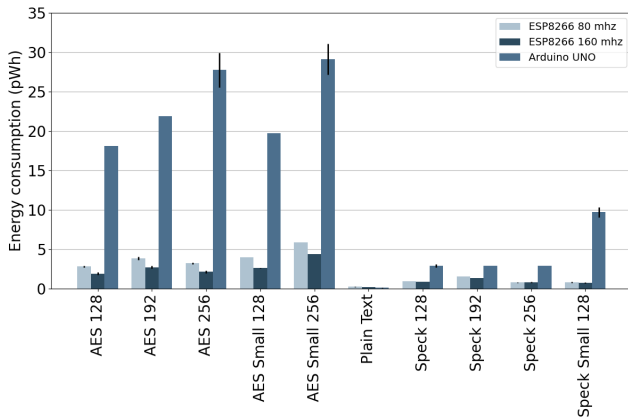
(b) Receiver, decryption.

Figure 2. Time to encrypt and decrypt. The Speck family outperforms the AES family, with significant time differences for both hardware platforms.

less energy than AES, as expected, but the key size has negligible influence on the energy consumption. In Figure 3(b), it is observed that the lowest decryption consumption estimate is also obtained for the Speck 128 algorithm when using Arduino UNO. In the AES family, the lowest estimate occurs for the traditional algorithm with a 128-bit key in Arduino UNO. The WeMos D1 ESP8266 consumes less energy when using the Speck family for decryption. The 192-bit key size consumes slightly more energy than the other key sizes. The same behavior is observed for the AES algorithm. It should be noted that the results for the Speck Tiny 128 and AES Tiny 128 algorithms are omitted because both do not implement a decryption function. The result demonstrates the energy efficiency of the Speck algorithm for encrypting data. It is interesting to highlight that algorithms optimized for memory usage increase energy consumption compared to the original implementations. Unexpectedly, in both encryption and decryption, the WeMos D1 ESP8266 with higher CPU frequency consumes less energy when using the AES algorithm. This may occur because the time needed to encrypt or decrypt using AES is lower for the operation mode in 160 MHz. For the Speck algorithm, the CPU frequency does not influence the result. Since the estimated consumption is extremely low, in the order of picowatt-hours, any of the algorithms could be applied in an energy-constrained



(a) Transmitter, encryption.



(b) Receiver, decryption.

Figure 3. Estimated energy consumption. All algorithms exhibit low consumption, with the lowest average consumption obtained for the Speck family, considering both microcontrollers for transmission and reception. The WeMos D1 ESP8266 is more energy efficient, outperforming the Arduino UNO.

system. However, if it is necessary to minimize consumption, the ideal algorithm is Speck with a 128-bit key for systems based on Arduino UNO, or Speck 128/256 for WeMos D1 ESP8266-based systems. Considering the lowest capacity battery commonly used in IoT, the CR2032 with 3 V and 225 mAh (Smith *et al.* [2020]), the Arduino UNO transmitter's lifetime would be approximately 578×10^6 years, and the receiver's 786×10^6 years. Considering only the use of encryption, the traditional Speck algorithms reduce the Arduino UNO transmitter's lifetime to 0.04% of the original, and the receiver's to 3.4%. In the transmitter, the memory-optimized versions and the AES family drop to 0.01%. In the receiver, the Speck Small 128 version drops to 1%, while AES 128 and AES Small are around 0.5%, AES 192 and AES 256 at 0.4%, and AES Small 256 at 0.3%. For the WeMos D1 ESP8266 transmitter, the lifespan without cryptography would be 193×10^6 years, and the receiver would have a lifespan of 332×10^6 . The shorter lifespan compared to the Arduino UNO is due to the significantly slower process for transmitting and receiving plain text using the WeMos D1 ESP8266. Considering only the encryption, the traditional Speck algorithms reduce the transmitter's and receiver's lifetime of the WeMos D1 ESP8266 microcontroller operating at 80 MHz to 0.25% of the original. The less energy-hungry AES algorithm when the WeMos D1 ESP8266 operates at 80 MHz is the traditional AES with a 128-bit key. This algorithm reduces the lifetime of the transmitter to 0.13% and the receiver to 0.08% of the original.

Regarding memory usage, both flash memory and SRAM are evaluated to check the impact of the algorithms on the available storage space in the devices. This usage refers to the entire code, including initialization, encryption, and transmission in the transmitter, and initialization, reception, and decryption in the receiver. Figure 4 shows the obtained results as a percentage of memory occupancy for each type of memory. Flash memory usage tends to be lower for the Speck algorithm family when used in the Arduino UNO R3, being minimal for the Speck Tiny 128 in the transmitter and Speck 128/192/256 in the receiver. For the AES family, the lowest usage is obtained for the AES Tiny algorithm with a 128-bit key in the transmitter and AES 128/192 in the receiver. Considering the WeMos D1 ESP8266, the flash memory occupancy is almost constant across all algorithms and does not differ much from the occupancy of an implementation that transmits only plain text. Considering SRAM memory, usage is minimized in the Arduino UNO transmitter when using the Speck Tiny algorithm with a 128-bit key. The Speck Small algorithm with a 128-bit key shows the second lowest memory usage both in the transmitter and the receiver. The worst performance in terms of SRAM memory consumption is observed for the AES algorithm with a 256-bit key. Among the AES family, the algorithm with the best performance is AES Tiny with a 128-bit key. In turn, the SRAM occupancy for the WeMos D1 ESP8266 transmitter and receiver is quite constant independently of the used algorithm, even for the optimized versions of Speck and AES. Considering that the Tiny variations do not have their own decryption function, in scenarios with restricted computational resources based on Arduino UNO, it is more advantageous to use the Speck Small algorithm. For scenarios based on We-

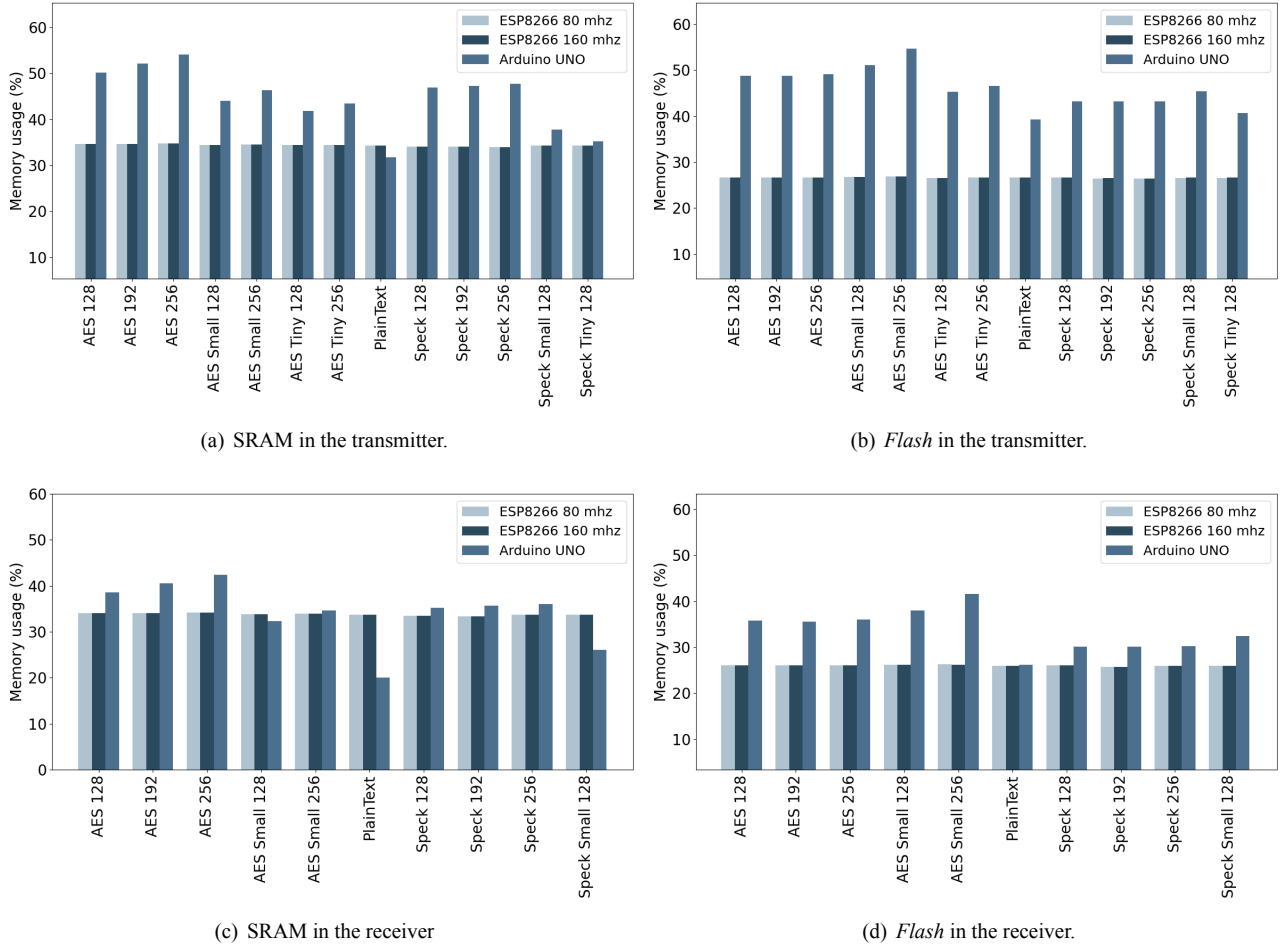


Figure 4. Total memory usage. The lowest usage occurs for Speck Tiny 128 in the transmitter and Speck Small 128 in the receiver. The usage for these algorithms is similar to that of the code without encryption (PlainText).

Mos D1 ESP8266, the algorithm is indifferent. It is worth highlighting that the flash and SRAM memory usage for the Speck Small 128 algorithm using the Arduino UNO is very close to the usage observed in the scenario without encryption (PlainText), corroborating the memory optimization of the algorithm. This optimization, however, is not noteworthy for the WeMos D1 ESP8266.

Table 1 summarizes the best algorithms for each evaluated metric, considering both the transmitter (Tx.) and receiver (Rx.), using the Arduino UNO R3 (Δ) and the WeMos D1 ESP8266 (\bullet) boards. The throughput is algorithm- and hardware-indifferent, considering the hardware platforms evaluated in this paper. In terms of energy consumption and execution time, the Speck family stands out. The variation in memory consumption considering all algorithms for the WeMos D1 ESP8266 is insignificant. Nevertheless, for the more resource-constrained Arduino UNO R3, the optimized Speck algorithms consume less SRAM, but the traditional implementation consumes less flash memory. It is clear from these results that resource-constrained devices benefit from lightweight cryptographic algorithms. Moreover, for highly resource-constrained devices, such as the Arduino UNO R3, the benefit is more significant, as the lightweight algorithms consume less memory, allowing the developer to implement more features to be executed by the microcontroller. The results underscore that in the development of non-critical ap-

plications, concerns regarding the impact of cryptographic algorithms on performance are minimal. In such cases, the primary focus should be on the security features provided by the algorithms. On the other hand, for critical applications, the careful selection of a suitable algorithm is paramount, particularly when utilizing resource-constrained devices, as an inappropriate choice could compromise the system operation.

6 Conclusions

This work evaluated the performance of symmetric encryption algorithms on resource-limited hardware to enable the construction of secure remote monitoring IoT systems. The evaluation was carried out through practical experiments using Arduino UNO R3 and WeMos D1 ESP8266 transmitters and receivers, in which the AES and Speck algorithms were implemented in their original and memory-optimized versions, varying the key size. The results revealed that in terms of energy consumption, the traditional Speck algorithm with a 128-bit key stands out, being also the algorithm with the shortest execution time. When running a less constrained device, such as the WeMos D1 ESP8266, the key size for the Speck family has negligible influence. The SRAM and flash memory consumption was lower for Speck Tiny 128,

Table 1. Summary of best results considering both hardware platforms to build the transmitter (Tx.) and the receiver (Rx.).

Algorithm	Microcontroller	Energy consumption		Execution time		SRAM consumption		Flash consumption		Throughput
		Tx.	Rx.	Tx.	Rx.	Tx.	Rx.	Tx.	Rx.	
Speck 128	Arduino UNO R3	△	△	△	△				△	△
	WeMos D1 ESP8266	●	●			●	●	●	●	●
Speck 192	Arduino UNO R3		△	△	△				△	△
	WeMos D1 ESP8266	●		●	●	●	●	●	●	●
Speck 256	Arduino UNO R3		△	△	△				△	△
	WeMos D1 ESP8266	●	●	●	●	●	●	●	●	●
Speck Small 128	Arduino UNO R3						△			△
	WeMos D1 ESP8266		●			●	●	●	●	●
Speck Tiny 128	Arduino UNO R3					△		△		△
	WeMos D1 ESP8266					●	●	●	●	●
AES 128	Arduino UNO R3									△
	WeMos D1 ESP8266					●	●	●	●	●
AES 192	Arduino UNO R3									△
	WeMos D1 ESP8266					●	●	●	●	●
AES 256	Arduino UNO R3									△
	WeMos D1 ESP8266					●	●	●	●	●
AES Small 128	Arduino UNO R3									△
	WeMos D1 ESP8266					●	●	●	●	●
AES Tiny 128	Arduino UNO R3									△
	WeMos D1 ESP8266					●	●	●	●	●

followed by Speck Small 128 when Arduino UNO is used, but the occupancy is algorithm-indifferent for the WeMos D1 ESP8266. The memory-optimized implementations showed worse energy performance. Due to the very short execution times, there was little influence on the transmission throughput, which reached 3.4 packets per second in the evaluated scenario. The WeMos D1 ESP8266 CPU frequency does not influence the performance of the Speck family. Future work envisions evaluating other cryptographic techniques, including cryptographic hash functions to ensure data integrity, and other lightweight algorithms, such as the Ascon family. Furthermore, the performance of the IoT gateway in receiving data from multiple sensors should be verified.

Declarations

Acknowledgements

The authors acknowledge the City Hall of Niterói for the insights to develop the remote monitoring system.

Funding

This research was partially funded by the City Hall of Niterói, CNPq, RNP, and FAPERJ.

Authors' Contributions

Mayksuel Ramalho developed and implemented the software used in this work, as well as performed the experiments and analysis. Gabriel de Oliveira contributed to the data curation, the writing of the original draft, and the experimental methodology. Nicholas Neves assisted in software development and experiments. Rafael Porto assisted in the data curation process. Victor Sobral assisted with data visualization. Marcos Rezende provided insights on monitoring systems and assisted in the supervision of this work. Dianne Medeiros contributed to the conceptualization, supervision, writing of the original draft, and review and editing of the final work.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The Arduino codes used in this work are available at <https://github.com/scherly/jisa-crypto>.

References

- Abdullah, A. M. (2017). Advanced encryption standard (AES) algorithm to encrypt and decrypt data. *Cryptography and Network Security*, 16(1):11. Available at: https://www.researchgate.net/publication/317615794_Advanced_Encryption_Standard_AES_Algorithm_to_Encrypt_and_Decrypt_Data.
- Albarelo, R., Oyamada, M., and de Camargo, E. (2020). Evaluation of cryptographic algorithms and implementation of a lightweight protocol for communication between IoT devices. In *Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, pages 65–72, Porto Alegre, RS, Brazil. SBC. (In Portuguese). DOI: 10.5753/sbesc_estendido.2020.13092.
- Alvalá, R. C. S. and Barbieri, A. F. (2017). Natural disasters. In *Mudanças climáticas em rede: um olhar interdisciplinar*, volume 1, pages 203–230. Canal6Editora. Available at: https://www.researchgate.net/profile/Jose-Marengo-2/publication/322638754_Mudancas_Climaticas_em_Rede_Um_olhar_interdisciplinar-Contribuicoes_do_Instituto_Nacional_de_Ciencia_e_Tecnologia_para_Mudancas_Climaticas/links/5a65d6efaca272a158200a2c/Mudancas-Climaticas-em-Rede-Um-olhar-interdisciplinar-Contribuicoes-do-Instituto-Nacional-de-Ciencia-e-Tecnologia-para-Mudancas-Climaticas.pdf#page=205.
- Beg, A., Al-Kharobi, T., and Al-Nasser, A. (2019). Performance evaluation and review of lightweight cryptography in an internet-of-things environment. In *2nd International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–6. DOI: 10.1109/CAIS.2019.8769509.
- Bogdanov, A., Khovratovich, D., and Rechberger, C. (2011). Biclique cryptanalysis of the full AES. In Lee, D. H. and

- Wang, X., editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 344–371, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-25385-0_19.
- de Paiva, B. D., de Souza, B. P., and Travassos, G. H. (2023). Cryptocomponent: a cryptography component for low cost IoT software systems. In *Anais Estendidos do XIV Congresso Brasileiro de Software: Teoria e Prática*, pages 90–99. SBC. (In Portuguese). DOI: 10.5753/cb-soft_estendido.2023.235935.
- El-hajj, M., Mousawi, H., and Fadlallah, A. (2023). Analysis of lightweight cryptographic algorithms on IoT hardware platform. *Future Internet*, 15(2). DOI: 10.3390/fi15020054.
- Fotovat, A., Rahman, G. M. E., Vedaiei, S. S., and Wahid, K. A. (2021). Comparative performance analysis of lightweight cryptography algorithms for IoT sensor nodes. *IEEE Internet of Things Journal*, 8(10):8279–8290. DOI: 10.1109/JIOT.2020.3044526.
- Guinelli, J. V., Aguiar, O. V., and Lazarin, N. M. (2018). Analysis and comparison of symmetric cryptographic algorithms embedded into arduino platform. In *Anais Estendidos do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 167–176, Porto Alegre, RS, Brazil. SBC. (In Portuguese). DOI: 10.5753/sbseg_estendido.2018.4153.
- Hossein Motlagh, N., Mohammadrezaei, M., Hunt, J., and Zakeri, B. (2020). Internet of things (iot) and the energy sector. *Energies*, 13(2). DOI: 10.3390/en13020494.
- Maitra, S., Richards, D., Abdelgawad, A., and Yelamarthi, K. (2019). Performance evaluation of iot encryption algorithms: Memory, timing, and energy. In *2019 IEEE Sensors Applications Symposium (SAS)*, pages 1–6. DOI: 10.1109/SAS.2019.8706017.
- Mouha, R. A. and Ait, R. (2021). Internet of things (IoT). *Journal of Data Analysis and Information Processing*, 09(2):77–101. DOI: 10.4236/jdaip.2021.92006.
- Munoz, P. S., Tran, N., Craig, B., Dezfouli, B., and Liu, Y. (2018). Analyzing the resource utilization of aes encryption on iot devices. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1200–1207. DOI: 10.23919/APSIPA.2018.8659779.
- Panahi, P., Bayılmış, C., undefinedavuşoğlu, U., and Kaçar, S. (2021). Performance evaluation of lightweight encryption algorithms for IoT-based applications. *Arabian Journal for Science and Engineering*, 46(4):4015–4037. DOI: 10.1007/s13369-021-05358-4.
- Pereira, G. C. C. F., Alves, R. C. A., Silva, F. L. d., Azevedo, R. M., Albertini, B. C., and Margi, C. B. (2017). Performance evaluation of cryptographic algorithms over IoT platforms and operating systems. *Security and Communication Networks*, 2017:1–16. DOI: 10.1155/2017/2046735.
- Qasaimeh, M., Al-Qassas, R. S., and Ababneh, M. (2021). Software design and experimental evaluation of a reduced aes for iot applications. *Future Internet*, 13(11). DOI: 10.3390/fi13110273.
- Sleem, L. and Couturier, R. (2020). Speck-r: An ultra lightweight cryptographic scheme for internet of things. *Multi-media Tools and Applications*, 80(11):17067–17102. DOI: 10.1007/s11042-020-09625-8.
- Smith, R., Palin, D., Iouliauou, P. P., Vassilakis, V. G., and Shahandashti, S. F. (2020). Battery draining attacks against edge computing nodes in IoT networks. *Cyber-Physical Systems*, 6(2):96–116. DOI: 10.1080/23335777.2020.1716268.
- Song, L., Huang, Z., and Yang, Q. (2016). Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. In Liu, J. K. and Steinfeld, R., editors, *Information Security and Privacy*, pages 379–394, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-40367-0_24.
- Tawalbeh, L., Muheidat, F., Tawalbeh, M., and Quwaidar, M. (2020). IoT privacy and security: Challenges and solutions. *Applied Sciences*, 10(12). DOI: 10.3390/app10124102.
- Vaz, Y., Mattos, J., and Soares, R. (2023). Optimized AES for use in IoT applications. In *Anais Estendidos do XIII Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, pages 31–36, Porto Alegre, RS, Brazil. SBC. (In Portuguese). DOI: 10.5753/sbesc_estendido.2023.235422.
- Zanon, V., Romancini, E. M., Manoel, B., Lau, J., Ourique, F., and Morales, A. (2022). Experimental evaluation of a security layer implemented in a cardiac wearable device for the medical internet of things. In *Anais do XXII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 97–110, Porto Alegre, RS, Brazil. SBC. (In Portuguese). DOI: 10.5753/sbseg.2022.224659.