A Comparative Analysis of the Impact of Cryptography in IoT LoRa Applications

Inês Lino
LASIGE, Faculdade de Ciências
Universidade de Lisboa
Lisboa, Portugal
ineslino07@gmail.com

José Cecílio
LASIGE, Faculdade de Ciências
Universidade de Lisboa
Lisboa, Portugal
jmcecilio@ciencias.ulisboa.pt

2022 IEEE 20th International Conference on Industrial Informatics (INDIN) | 978-1-7281-7568-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/INDIN51773.2022.9976108

***Abstract*—The Internet of Things has been applied in different application contexts, ranging from industrial, healthcare or simple home life applications. It allows adding a new level of automation to objects, making them appealing to everyone. Since several applications involve data, and in some cases, it is sensitive, those applications are more vulnerable to be attacked. As a result, researchers are constantly exploring secure mechanisms to keep the data and its owners safe. For the resource-constrained nature of these IoT applications, the design of cryptography solutions becomes a challenging task because IoT nodes should be cheap and require low power, which means lower computational performance. Lightweight cryptography algorithms are attractive solutions to reduce computation complexity, keeping the desired level of security. This work provides an experimental performance analysis of several cryptography solutions targeted for embedded platforms to help choose better algorithms for IoT systems in terms of efficiency. Execution time, code efficiency and communication influences are evaluated for a set of cryptography algorithms. The results of this work intend to help IoT developers to choose a suitable cryptography algorithm for their IoT applications.**

***Index Terms*—IoT applications, Lightweight cryptography, Communication security, Performance evaluation**

I. Introduction

With the evolution of technology, mainly due to the capabilities of devices, access to information is increasingly smart.

In an IoT environment, there are several restricted devices connected to the Internet. These devices interact with each other over a network, providing us with an extensive range of services and new experiences. For us to enjoy this new environment, the security of the devices is critical. If a device is compromised, the network can suffer serious consequences (such as eavesdropping or denial of service). It is not easy to implement sufficient secure cryptographic functions on devices with restrictions such as the ones used in IoT environments. Typically, those devices have limited resources regarding computation capabilities, memory and energy that can be dropped during (heavy) cryptography processes.

E. Bertino and N. Islam in [1] discussed the vulnerabilities in IoT systems, where they mentioned, in the first place, the lack of authentication or authorization mechanisms. Besides this limitation, they said that most IoT applications offer insecure network services, do not provide transport encryption and integrity verification mechanisms. Additionally, the authors also mentioned that, in many cases, the Software and firmware are insecure, and there are no management systems for updating it remotely.

It is worth noting that fixing security issues in the IoT is more challenging than on the Internet. Its inherent features, such as connecting physical actions with remote communications through various devices and sensors, make it more challenging.

There are many trusted encryption algorithms for the Internet, e.g., advanced encryption standards (AES) and Rivest–Shamir–Adleman (RSA). However, there were generally designed for servers, smartphones, and desktops, where a considerable amount of computational resources is available [2], which means that these ciphers might not be the right choice for IoT nodes.

For IoT applications, encryption algorithms that require small resources are needed. In [3], an hardware implementation over constrained devices is discussed. The authors said that the performance could be defined as the required gate area, the number of logic blocks, or gate equivalents. In software implementations, the performance of these algorithms is measured by the usage of the random access memory (RAM), the read-only memory, and the central processing unit (CPU) [2], [4]. Power consumption, latency, and throughput are other important metrics that need to be considered in the trade-off between the desired level of security and the resources of IoT nodes. Since many limitations may be considered, there is no general agreement on IoT networks' encryption/authentication algorithms.

The National Institute of Standards and Technology (NIST) has a task force running to design cryptographic algorithm patterns that can work within the limits of a simple IoT device, where memory, battery, or real-time response restrictions are presented. A vast number of encryption ciphers (e.g., ChaCha20, CLEFIA, TWINE, LED, SPECK, SIMON, PICCOLO, PRESENT, ANU) have been proposed considering the lightweight encryption requirements [5], as well as some well-known algorithms have been optimized to fit in the world of lightweight cryptography, like AES and DES [6]. Some of these algorithms are ultra-lightweight ciphers, convenient to

AQUAMON research project, ref. PTDC/CCI-COM/30142/2017 and the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020.

extremely constrained environments such as RFID tags.

For IoT applications, we need to use encryption algorithms that require small resources. In [5], authors discuss experimental results of the performance of AES, Twofish, Speck, RC6, and LEA with different key sizes. Maitra et al., in [7], did a study that evaluates the required timing, memory, and power for software implementation of AES-256 and XTEA ciphers on 8-b and 32-b microcontroller-based systems. In [8], the power consumption of AES, and RSA ciphers for encryption and decryption is evaluated. Singh et al., in [9], propose a method to assign an encryption algorithm to each IoT device based on memory, computation power and message data size.

In this work, we explore the applicability of several cryptographic algorithm ciphers to the real context of the Internet of Things (IoT). The performance of the algorithms is investigated and compared. All tested ciphers are in authenticated encryption with associated data (AEAD) format, which takes plain text, associated data, nonce, and key as inputs. We provide experimental performance analysis of several lightweight algorithms to help choose better algorithms for IoT applications in terms of efficiency. Unlike the previous simulation-based benchmark works, we used 32-bit ARM (Raspberry Pi 3) microcontrollers to run real experiments.

II. Cryptography patterns

Cryptography is divided in two types, symmetric and asymmetric encryption. Symmetric encryption ciphers use a single shared key. This key must be shared among the nodes who need to receive the message, while asymmetric encryption ciphers use a pair of keys, one public and one private used to encrypt and decrypt messages, respectively.

Regarding symmetric cryptography, the algorithms behind this approach can be divided into two categories, based on its input type: block ciphers and stream ciphers.

A. Block ciphers

A block cipher is an encryption approach that requires a fixed input size of x bits and produces a ciphertext of also x bits. Furthermore, if the input is more extensive than x bits, it is divided into several blocks. There are several modes of operations for a block cipher that fit different applications. For instance, the Electronic Code Book (ECB) mode uses simple substitution, where the input plaintext is broken into several blocks and encrypted individually using the key [10].

The Cipher Block Chaining (CBC) is a confidentiality mode whose encryption process features the combining ("chaining") of the plaintext blocks with the previous ciphertext blocks given as input to the following encryption algorithm after applying an XOR operation with the original plaintext block [11], [12].

The Cipher Feedback (CFB) mode is another confidentiality mode that uses the feedback of successive ciphertext segments into the input blocks. While the Counter (CTR) mode uses counters to produce a sequence of output blocks [11].

The AES-XTS Block Cipher Mode (XTS) is a block cipher mode designed to be a more robust alternative to other available block cipher modes. It eliminates potential vulnerabilities associated with some of the more sophisticated side-channel attacks used to exploit weaknesses within other methods [13].

Additionally, block ciphers can be classified accordingly to their design structure [14]. These design structures can be Feistel ciphers, Lai-Massey ciphers, and Substitution-Permutation networks. In a Feistel cipher, the block of plain text is split into two equal-sized halves. A round function is applied to one half, using a subkey, and then the output is XORed with the other half. The two halves are then swapped.

The Lai-Massey structure offers security properties like the Feistel structure. It also provides the advantage that the round function F does not have to be invertible. Another similarity comes from the division of the input block into two equal pieces. However, instead of applying the round function only to one part, the round function is computed based on the difference between the two parts in the Lai-Massey structure. The result is then added to both half blocks.

In the substitution-Permutation networks, a network takes a block of the plaintext and the key as inputs and applies several alternating "rounds" or "layers" of substitution boxes (S-boxes) and permutation boxes (P-boxes) to produce the ciphertext block. The S-boxes and P-boxes transform (sub-)blocks of input bits into output bits. These transformations are common for efficient operations to perform in hardware, such as exclusive or (XOR) and bitwise rotation. The key is introduced in each round, usually in the form of "round keys" derived from it. Decryption is done by reversing the process (using the inverses of the S-boxes and P-boxes and applying the round keys in reversed order).

B. Stream ciphers

In contrast to the block ciphers, stream ciphers are applied to single bits of data and a cryptographic key is used to generate a pseudo-random stream of digits combined with the plaintext digits to create the ciphertext. The sequence of (pseudo) random digits used in the process is called a "keystream". It has the same length as the plaintext message. The keystream is typically XOR with the plaintext using a bitwise operation. Stream ciphers provide faster encryption and decryption capability than block ciphers and can be used in symmetric and asymmetric key cryptosystems [10]. There are two types of stream ciphers, synchronous stream ciphers, where the keystream is generated independently of the plaintext and ciphertext messages, and asynchronous stream ciphers. Asynchronous stream ciphers are also known as self-synchronizing stream ciphers (SSSC). These ciphers can self-correct the output stream when a set of limited error ocur in the transmission. When a recipient detects an error, an automatic re-synchronizing process with the keystream generator occurs.

Taking into consideration all patterns, in Figure 1 we depict a taxonomy of the main cryptography algorithms that are considered in this work. We call the readers' attention that RSA, DES, and AES algorithms were not considered specifically to be used in the IoT context or even for lightweight cryptography. However, since they are widely used in traditional

cryptography approaches, we use them as a reference for the comparison done in this work.

III. Performance Analysis

The presented work investigates and compares the performance of several lightweight Cryptography algorithms in 32bit embedded platforms used to develop IoT applications. The code size, execution time, efficiency for the algorithms over three different hardware platforms using different message sizes are evaluated. Since there are many algorithms, we concentrate our evaluation on a sub-set of them. We will evaluate and compare, at least, one algorithm of each category and approach defined in Figure 1. We will evaluate the following algorithms: DES [15], PRESENT [16], PICCOLO [17], IDEA [18], AES [19], CLEFIA [20], SALSA20 [21], XCHACHA20 [22], SHA [23], MD5 [24] and RSA [25].

A. Evaluation setup

Three hardware platforms were chosen for the evaluation. The hardware was selected from the popular platforms, considering high, medium, and low resource availability with the capability to run C/C++ code. All of the chosen platforms include a LoRA chip used to send data within a LoRa network. Each Raspberry Pi is equipped with a Dragino shield, which provides that capability and the Arduino MKR1300 already includes a LoRa chip. Table I shows the main features of each hardware platform.

TABLE I: Hardware Environment

Platform Name	**Microcontroller Core**	Frequency	RAM
RPI-3B	ARM A53 Quad Core	1.2 GHz	1GB
RPI-ZW	ARM 1176 Single Core	1 GHz	512MB
ARDUINO MKR1300	SAMD21 Cortex M0+	48 MHz	32 KB

Raspbian was selected as the Linux OS for the Raspberry Pi 3B (RPI-3B) and Raspberry Pi Zero W (RPI-ZW). The GCC was used, in Raspbian, to compile the code without any memory optimization. The Arduino IDE is used to compile and upload the code for the Arduino MKR1300 platform. Monitoring software was written in C language to monitor the execution times, the CPU and RAM usage, which then stored the data in the SD-card to be later processed and visualized. The serial monitor tool, included in the Arduino IDE, was used to collect the monitoring data concerning the Arduino platform. Both the Raspberry Pi platforms are controlled and monitored using a remote personal computer (PC) through SSH, and the Arduino is directly connected to the PC through the serial port.

B. Results and Discussion

To evaluate and compare the different algorithms, an exhaustive performance analysis was done over a specific set of algorithms. For this evaluation, one or two algorithms were chosen from the various cryptography schemes. From the Symmetric Key Cryptography, for the Block Cipher scheme, using the Feistel Cipher, we evaluate the DES, PRESENT and PICCOLO algorithms. In terms of Lai-Massey structure and Substitution-Permutation networks, we consider the IDEA, and AES and CLEFIA, respectively. In terms of Stream Cipher, the SALSA20 and XCHACHA are evaluated. Related to Hashing Techniques, the SHA and MD5 are also tested. Concerning Asymmetric Key Cryptography, we try the RSA algorithm. Since this algorithm is widely used in computer networks, we tested it in this context. However, it may not appropriated for IoT applications. Figure 2 shows the code size (the amount of ROM memory) used by each algorithm. Depending on the complexity of each algorithm, the ROM memory needed is different. All algorithms were developed in the C/C++ programming language. Both Raspberry PI platforms run the Raspbian Linux OS, making them available to run the exact implementation of the algorithms. For the Arduino MKR1300, we did a set of modifications to fit the platform and IDE specifications. For instance, we avoid using dynamic memory allocation and use static data structures to implement the algorithms. Due to this modification, the memory required for some algorithms is less when compared with the Raspberry PI implementation.

Once we developed all algorithms, we did some experiments to evaluate the execution time of each algorithm. All algorithms were tested by running them five times when operating on 100 Bytes of data and with a key of 128-bits. In order to avoid any interference due to external sensors, all data is randomly generated internal to the testing platform. Figure 3 shows the results concerning this experiment. As expected, the Arduino MKR1300 presents the worst results. It has a reduced clock which makes it slower to execute the different operations required by the algorithms. However, due to its low resources, it is a cheaper platform, and it is widely used to build IoT systems. Both implementations show a good performance and follow the same patterns when compared between different algorithms in terms of the Raspberry PI implementation. In this case, the RPI-ZW offer less performance, but this was expected because it has fewer resources than the RPI-3B.

In Figure 3, we can see, for the Raspberry PI implementation, that DES and XCHACHA with a 128-bit key were the algorithms that are able to process more bytes per second. It can reach almost 1.5 MB/s. Looking at the Arduino performance, the DES and the MD5 algorithms were the best, allowing them to get 500 kB/s. During this experiment occurred some outliers that we neglect. They might be due to randomly generated data or CPU availability when the experiment ran.

Figure 4 shows the influence of data size in the execution times. From the Figure, we can see a direct relationship between the data size and the execution time. The most unpredictable case appears in the Arduino MKR 1300 platform. In this case, we can see a direct relation, but it is not true for others. Concerning the same platform, it was unable to run

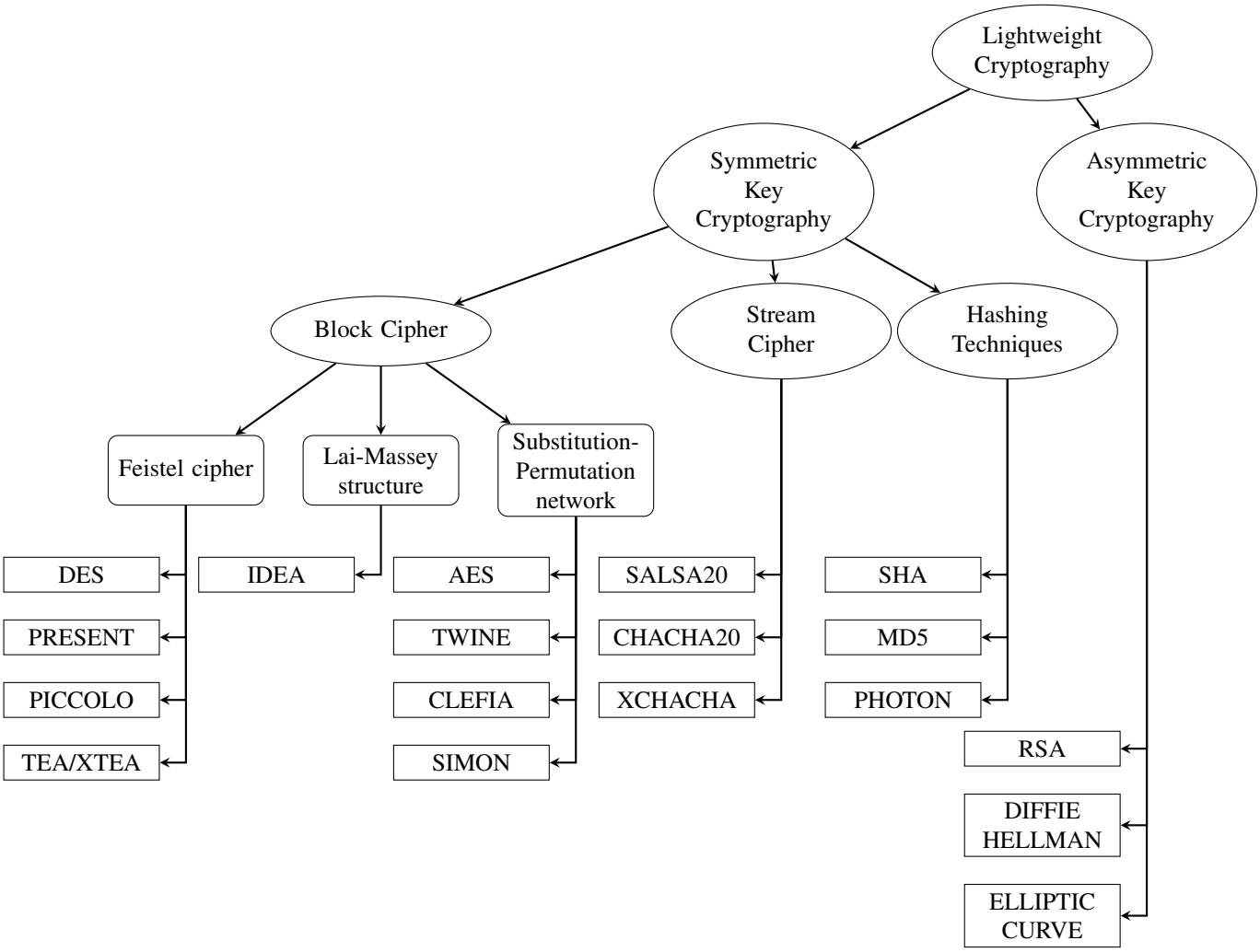


Fig. 1: Taxonomy of Cryptographic algorithms.

most algorithms when 500 Bytes data was tested.

To evaluate the influence of cryptography on communication, we did a set of simulations to understand the influence that a cryptography algorithm may introduce in a LoRa network. Since the message size produced by the cryptography algorithms may change, depending on its configurations and the data itself, in Figure 6 we show the Time on Air (ToA) required to transmit a message using a LoRa chip.

The values shown in Figure 6 take into account the packet structure of LoRa [26]. These values are calculated through Equation 1, where the SF represents the spreading factor. It can assume a value from SF5 to SF12. The BW is the Bandwidth, which determines the width of the transmitted signal and, consequently, the chirp duration. For this experiment, a bandwidth of 125KHz is considered.

$$T_{packet} = \frac{2^{\text{SF}}}{BW} * (Preamble_{Size} + PL_{Size}) \qquad (1)$$

The parameter $Preamble_{Size}$ represents the number of preamble symbols. It is defined by the user. However, it must include the 2 mandatory synchronization symbols and 2.25 symbols of Start Frame Delimiter. Lastly, the PL_{Size} represents the number of payload symbols. This parameter is the one that can be affected by applying a cryptography algorithm over data. It is determined according to the specification done in [26] and the work did in [27], which takes into account the data size and other parameters such as, the specification of an header or the CRC code used for the transmission. Since the communication range of LoRa depends on the SF, in Figure 6 we represent the ToA variation according to the data size and the SF. From this Figure, and mainly when an SF equal 12 is used, any increment on the data size introduces a significant variation on the amount of time needed to deliver a message. For this case, the selection of the cryptography algorithm should consider the output length produced by applying the algorithm. For the other cases, a slight increase in the output length is not very expressive in terms of ToA.

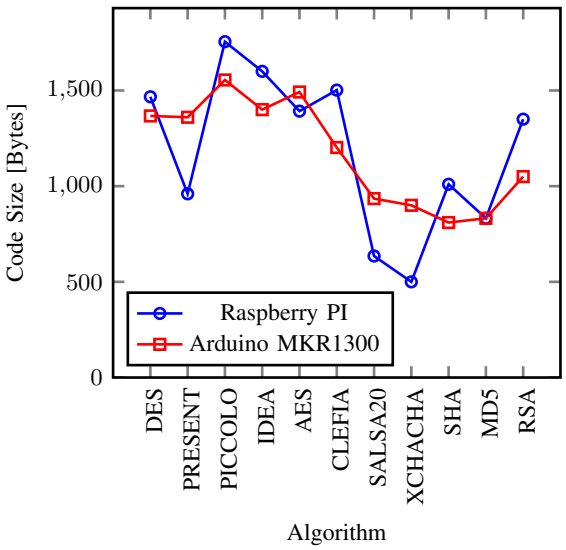


Fig. 2: Code Size.

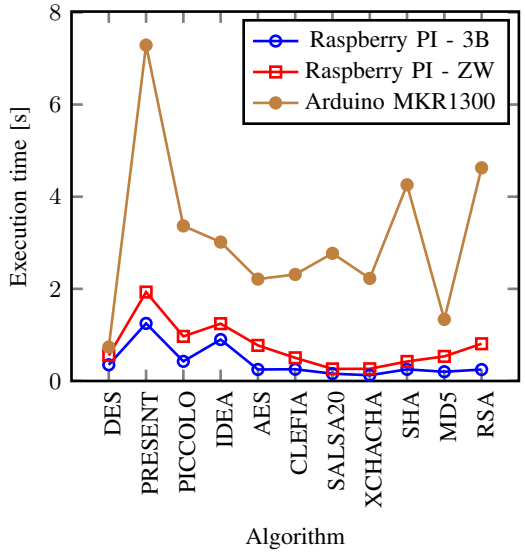


Fig. 3: Execution time for 100 Bytes of data.

IV. Conclusion

This article presented an experimental study where several cryptography algorithms, mainly those considered for lightweight cryptography, were implemented on three IoT node platforms. The platforms chosen can be used as a sensor or edge nodes of an IoT ecosystem. Parameters, such as the execution time, efficiency, data sizes and transmission time, were monitored in different test scenarios. The execution time varies significantly, depending on the algorithm, the data size and the platform where the algorithm runs. For example, considering the DES and PRESENT algorithms, these two algorithms are categorized as block chiper algorithms. They show a high difference in the execution time. This difference is emphasized if a platform with fewer computation resources was used.

Additionally, a practical application of the cryptography algorithm was simulated on an IoT sensor node using LoRa communication. The experiment showed that the encryption time required by the algorithms is more or less equal to the time needed to transmit the data, which can duplicate the amount of time necessary to get the sensor data at the gateway. Therefore, a user needs to carefully consider these factors while building an IoT system. This work can assist researchers in evaluating the pros and cons of different cryptography approaches (e.g., block cipher versus stream cipher) and developing new IoT applications taking into account security requirements.

Acknowledgment

This work was supported by FCT through funding of the AQUAMON research project, ref. PTDC/CCI-COM/30142/2017 and the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020.

References

[1] Elisa Bertino and Nayeem Islam. Botnets and internet of things security. *Computer*, 50(2):76–79, 2017.

[2] Meltem Sönmez Turan, Kerry McKay, Donghoon Chang, Cgdas a Calık, Lawrence Bassham, Jinkeon Kang, and John Kelsey. Status report on the second round of the nist lightweight cryptography standardization process. 2021.

[3] Xuyang Ding, Xiaoxiang Wang, Ying Xie, and Fagen Li. A lightweight anonymous authentication protocol for resource-constrained devices in internet of things. *IEEE Internet of Things Journal*, 2021.

[4] Ahmad Mustafa Mohamad Al-Aboosi, Samar Kamil, Siti Norul Huda Sheikh Abdullah, and Khairul Akram Zainol Ariffin. Lightweight cryptography for resource constraint devices: Challenges and recommendation. In *2021 3rd International Cyber Resilience Conference (CRC)*, pages 1–6. IEEE, 2021.

[5] Daniel AF Saraiva, Valderi Reis Quietinho Leithardt, Diandre de Paula, Andre Sales Mendes, Gabriel Villarrubia González, and Paul Crocker. Prisec: Comparison of symmetric key algorithms for iot devices. *Sensors*, 19(19):4312, 2019.

[6] Muzafer H Saračević, Saša Z Adamović, Vladislav A Miškovic, Mohamed Elhoseny, Nemanja D Maček, Mahmoud Mohamed Selim, and K Shankar. Data encryption for internet of things applications based on catalan objects and two combinatorial structures. *IEEE Transactions on Reliability*, 2020.

[7] Sudip Maitra, Dylan Richards, Ahmed Abdelgawad, and Kumar Yelamarthi. Performance evaluation of iot encryption algorithms: Memory, timing, and energy. In *2019 IEEE Sensors Applications Symposium (SAS)*, pages 1–6. IEEE, 2019.

[8] Joseph C Pry and Richard K Lomotey. Energy consumption cost analysis of mobile data encryption and decryption. In *2016 IEEE International Conference on Mobile Services (MS)*, pages 178–181. IEEE, 2016.

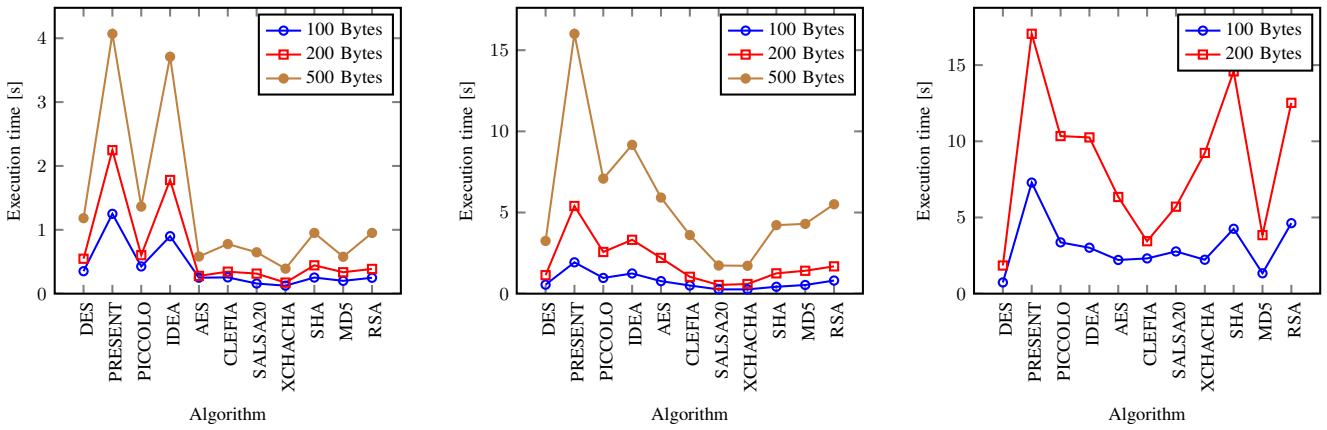
[9] Saurabh Singh, Pradip Kumar Sharma, Seo Yeon Moon, and Jong Hyuk Park. Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–18, 2017.

[10] Dale Liu. *Next generation SSH2 implementation: securing data in motion*. Syngress, 2011.

[11] Nicky Mouha, Morris Dworkin, et al. Review of the advanced encryption standard. Technical report, National Institute of Standards and Technology, 2021.

[12] Morris Dworkin. Recommendation for block cipher modes of operation. methods and techniques. Technical report, National Inst of Standards and Technology Gaithersburg MD Computer security Div, 2001.

[13] Dewi Laksmiati. Implementasi full disk encryption dengan algoritma aes-xts menggunakan veracrypt. *Jurnal Akrab Juara*, 4(3):1–10, 2019.



(a) RPI-3B. (b) RPI-ZW. (c) Arduino MKR1300.

Fig. 4: Execution time taking into account different data sizes.

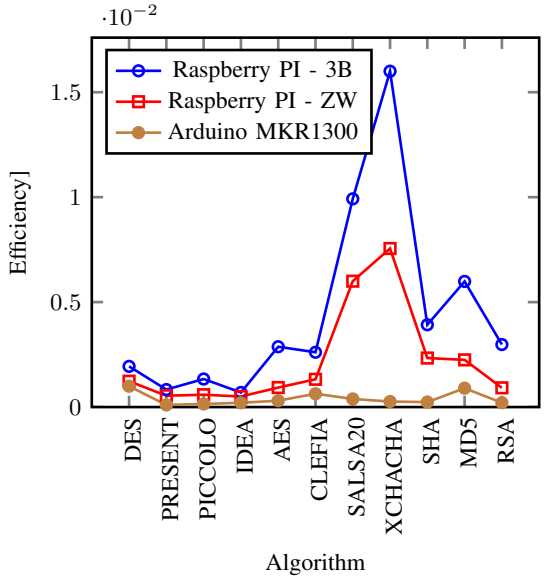


Fig. 5: Algorithm efficiency.

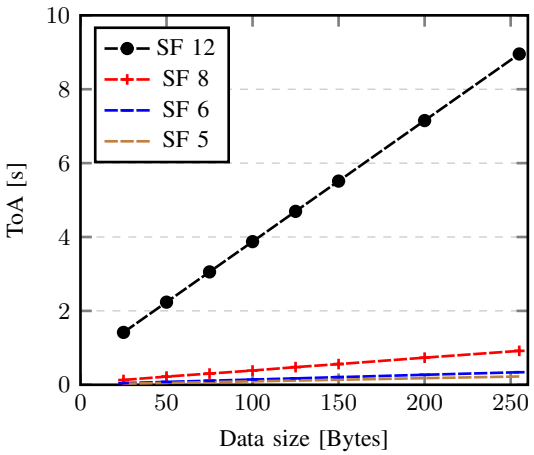


Fig. 6: Time on Air (ToA) according to different data sizes.

[14] Ragab Ahmed Ab M, Ahmed Madani, AM Wahdan, and Gamal MI Selim. Design, analysis, and implementation of a new lightweight block cipher for protecting iot smart devices. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–18, 2021.

[15] Na Su, Yi Zhang, and Mingyue Li. Research on data encryption standard based on aes algorithm in internet of things environment. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 2071–2075. IEEE, 2019.

[16] Runa Chatterjee and Rajdeep Chakraborty. A modified lightweight present cipher for iot security. In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pages 1–6. IEEE, 2020.

[17] Gandu Ramu, Zeesha Mishra, Pulkit Singh, and Bibhuendra Acharya. Performance optimised architectures of piccolo block cipher for low resource iot applications. *International Journal of High Performance Systems Architecture*, 9(1):49–57, 2020.

[18] How-Shen Chang. International data encryption algorithm. *jmu. edu, googleusercontent. com, Fall*, 2004.

[19] Weize Yu and Selçuk Köse. A lightweight masked aes implementation for securing iot against cpa attacks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(11):2934–2944, 2017.

[20] Lampros Pyrgas and Paris Kitsos. A very compact architecture of clefia block cipher for secure iot systems. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pages 624–627. IEEE, 2019.

[21] Evangelina Lara, Leocundo Aguilar, Jesús A García, and Mauricio A Sanchez. A lightweight cipher based on salsa20 for resource-constrained iot devices. *Sensors*, 18(10):3326, 2018.

[22] Fabrizio De Santis, Andreas Schauer, and Georg Sigl. Chacha20-poly1305 authenticated encryption for high-speed embedded iot applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 692–697. IEEE, 2017.

[23] Raffaele Martino and Alessandro Cilardo. Designing a sha-256 processor for blockchain-based iot applications. *Internet of Things*, 11:100254, 2020.

[24] K. Quist-Aphetsi and M. Xenya. Node to node secure data communication for iot devices using diffie-hellman, aes, and md5 cryptographic schemes. In *2019 International Conference on Cyber Security and Internet of Things (ICSIoT)*, pages 88–92. IEEE, 2019.

[25] Manuel Suárez-Albela, Paula Fraga-Lamas, and Tiago M Fernández-Caramés. A practical evaluation on rsa and ecc-based cipher suites for iot high-security energy-efficient fog and mist computing devices. *Sensors*, 18(11):3868, 2018.

[26] Semtech. Semtech SX1276, https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276. Accessed: 2022-04-05.

[27] José Cecílio, Pedro M Ferreira, and António Casimiro. Evaluation of lora technology in flooding prevention scenarios. *Sensors*, 20(14):4034, 2020.