International Journal of Information Security (2022) 21:1247–1262
https://doi.org/10.1007/s10207-022-00609-3

REGULAR CONTRIBUTION

Design and implementation of a new lightweight chaos-based cryptosystem to secure IoT communications

Abdenour Kifouche[1] · Mohamed Salah Azzaz[1] · Redha Hamouche[2] · Remy Kocik[2]

Published online: 1 September 2022

Abstract
Internet of things (IoT) provides several applications such as intelligent urban transportation, smart factory, and smart health. In such systems, the transmitted data are important and must be secured to prevent destructive and unauthorized access to critical data. Indeed, existing IoT solutions come with conventional cryptographic techniques such as AES, RSA, and DES. However, some of these algorithms are no longer reliable and others require significant resources in terms of energy, memory, and computing power, making them unsuitable for IoT nodes that may have limited resources. To address these challenges, this paper develops a new lightweight and efficient cryptosystem to secure IoT communications. The proposed cryptosystem is composed of a chaos-based random generator, confusion, and diffusion blocks. Through a use case, the experimental results, at implementation and statistical levels, demonstrate good performances. The implementation results in the Mbed microcontroller NXP LPC1768 include low memory usage, fast encryption and decryption speed, and low energy consumption. The statistical results confirm the robustness of the proposed cryptosystem against many attacks according to the NIST test, key size, key sensitivity, information entropy, and statistical histogram analysis. Compared to related works, this paper proposes an enhanced lightweight cryptosystem with optimized confusion–diffusion layers that can be implemented in different resource-constrained hardware boards. Moreover, the proposed solution does not make any assumptions about the data types to be used in IoT networks. It is open to any type (sensing value, text, voice, image, etc.). These features guarantee the potential use of the proposed cryptosystem in many real-world applications.

Keywords Internet of Things · Chaotic system · Chaos theory · Cryptosystem · Cryptography

1 Introduction

In the last decade, a large portion of the world's population has been living in cities. This portion is expected to increase to 70% by 2050 [1]. With the growth of cities, many problems appear such as pollution, traffic congestion, and waste management. Thus, monitoring and controlling these systems becomes increasingly difficult as their size, complexity, and interactions continue to grow. In addition, they are susceptible to frequent failures such as equipment failure, human error, and software error. In response to the urgency of addressing these issues, many initiatives are launched worldwide from city councils to companies and research laboratories [2]. Actors of various disciplines propose to connect these systems to the Internet to manage them in an efficient way, which led to the concept of the *Internet of Things*.

In IoT systems, it is becoming more and more obvious that transmitted data in such networks (Internet) are quite sensitive and have been susceptible to several attacks [3–5]. Recently, they have revealed a number of vulnerabilities, especially in smart transportation systems, smart grids, smart health systems, etc.

Smart transportation system has been found to be vulnerable. It allows remote control of the vehicle, as well as impersonation and sending false information to neighbor-

✉ Abdenour Kifouche
abdenour.kifouche@esiee.fr

Mohamed Salah Azzaz
ms.azzaz@gmail.com

Redha Hamouche
redha.hamouche@esiee.fr

Remy Kocik
remy.kocik@esiee.fr

[1] Laboratoire Systèmes Electroniques et Numériques, Ecole Militaire Polytechnique, Bordj El Bahri, Algeria

[2] ESIEE Paris, ESYCOM (UMR9007), Gustave Eiffel University, Noisy le Grand, France

ing vehicles [6]. Smart grid integrates different micro-grids via two-way communications between energy providers and consumers, depending heavily on reliable measuring data. Nevertheless, these systems have shown vulnerabilities to disruption of state estimation by data integrity attacks, erroneous data injection attacks, data deletion attacks, and data erasure attacks [7]. Smart health systems incorporate a number of sensors assessing the health status of patients and must conform to strict data privacy monitoring and regulatory requirements [8]. However, a number of health systems have been subverted, including viruses such as WannaCry, Medjack, and SamSam that have impacted hospitals [9,10].

Given these emerging threats, the security of systems and their vulnerability to various attacks affect all sectors and aspects [11]. Such risks require fast and efficient security algorithms. Indeed, existing IoT solutions come with security mechanisms at the transport and application layers. These mechanisms deploy conventional symmetric and asymmetric cryptography techniques such as AES, RSA, DES, and RC4. These algorithms are based on a permutation–diffusion network. However, some of them have been broken and are no longer trusted. Others (such as AES) require considerable resources for IoT nodes that may have limited resources in terms of energy, memory, and computational power [12].

The main problem raised in this work is related to the development of a dedicated encryption algorithm adapted to these nodes with limited resources. Indeed, devices with limited CPU, memory, and power resources are called *constrained devices* [13]. Based on the supported memory size, Seyhan et al. grouped them into three classes of constrained devices: C_0 for less than 10 KiB, C_1 for about 10 KiB, and C_2 for about 50 KiB [13]. IoT devices may also be limited in energy resources and their CPUs are characterized by low frequency. So, these devices have different resources and capabilities and are connected to the Internet. Thus, their data must be protected using encryption algorithms that are adapted to their limited resources. To deal with this challenge, several works proposed a cryptosystem adapted to these constrained devices calling them *lightweight cryptosystems*.

For the last decade, several lightweight algorithms for security were presented with different security levels to protect data through the communication channels between IoT nodes [14]. In order to obtain a relevant coverage of existing works related to lightweight cryptosystem, we performed queries. Going through the results of these queries, we only considered works dealing with IoT and lightweight cryptosystem concepts. In those works, the authors attempt to propose a "small" cryptographic algorithm, to face the very low battery load, the small size of ROM and RAM, and low CPU frequency. Some algorithms took advantage of the merits of the AES algorithm and attempted to adapt it to this constrained IoT node to ensure better security performances. Lee et al. [15] implemented the AES encryption algorithm on an 8-bit microcontroller as an example of a hardware device. They evaluate the performance of their implementation as a function of plaintext size and the cost of the operation per hop in a network. The encryption and decryption time are measured related to the data sizes of 16, 32, 64, 128, 256, and 512 Bytes. For a data size of 16 bytes, the measured encryption time is 449 ms, resulting in an encryption speed of 0.29 kbit/s. This value is not enough for some applications that require a high encryption speed. Omrani et al. [16] proposed a lightweight image cryptosystem for IoT devices called LICID. They implemented their solution in RaspberryPi. However, this board has significant resources, so it cannot be considered a constrained node. Abu Al-Haija et al. designed a microcontroller-based RSA [17]. They used the Arduino Mega2560R3 microcontroller to implement the proposed RSA coprocessor with a small key of 32 bits. Some works have proposed cryptosystems using VHDL and targeting FPGA boards. Aishwarya and Sreerangaraju proposed a secure and lightweight compressive sensing of stream cipher [18]. The proposed system is implemented using VHDL and simulated using the Modelsim tool. However, they did not give the resource consumption related to this implementation. Pasupuleti and Varma proposed a lightweight scheme with a Walsh–Hadamard transform access structure for providing data privacy in IoT [19]. Their concern is more about access polity and they did not implement their solution in constrained IoT nodes.

Recently, other works considered chaotic systems for building cryptosystems thanks to the good cryptographic properties of chaotic systems [20]. Indeed, chaotic systems have good cryptography features such as unpredictability, nonlinearity, aperiodicity, and high sensitivity to control parameters. In addition, their implementations require fewer resources in terms of processing, storage, and communication footprint compared to conventional approaches. Thus, they are lightweight in software development. These features made chaotic systems attractive for providing strong, lightweight, and efficient cryptography for resource-constrained IoT nodes.

Nguyen et al. present the design and the implementation of a low-power circuit for a chaos-based true random number generator and a chaos-key-based data encryption scheme for secure communications [21]. The chaotic system is realized at the hardware level using CMOS technology. Other approaches develop hardware solutions in FPGA. Those works differ from our concern which is more about a software solution that can be implemented in a constrained IoT node like a microcontroller. Nesa and Banerjee propose a chaos-based encryption algorithm built upon a discrete quadratic sinusoidal map [22]. The authors have simulated an IoT environment consisting of 4 IoT devices in MATLAB, each device generates data, encrypts it, and then sends it to other devices. Ahmad et al. develop a secure real-time scheme

for IoT systems by using an Intertwining Logistic chaotic map [23]. In [22,23], authors did not implement their solution in a constrained node. Akgul et al. focus on security between two nodes by using three different chaos generators [24]. This chaos-based encryption method is proposed to encrypt text data. Rajendran and Doraipandian propose a chaotic security architecture for ensuring the security of the medical images [25]. The authors of [26] have implemented their cryptosystem for GPS (Global Positioning System) data encryption. Those chaotic systems are proposed to encrypt specific data and are not adapted to different types of IoT applications. The closest article to our approach is proposed by Guerrero et al. [27]. They implemented a lightweight cryptosystem for constrained IoT nodes. Their algorithm is based on a chaotic system. The proposed approach is implemented on a PIC microcontroller. Compared to this work, we propose new confusion and diffusion algorithms that require fewer implementation resources and present better performances. In addition, in [27], memory cost, encryption speed, and power consumption were not discussed. Recent work has proposed a lightweight encryption algorithm, but without performing a full security analysis and without providing the implementation results [28,29].

Concluding this review of existing work, it is clear that these cryptosystems are designed for specific types of data such as text, images, or video data, while the trend in IoT applications is more orientated toward multiple data types such as sensing data from monitoring sensors in environmental fields, medical sensors, and other types of applications. In addition, some papers have used simulations to evaluate the effectiveness and strength of their cryptosystem. Others consider powerful IoT nodes in terms of resources. We believe that the implementation of encryption algorithms on real constrained IoT nodes provides more realistic results. For the other implementation, we also highlight the absence of some important implementation results such as memory usage, encryption speed, and power consumption. Some papers focus only on the generation of chaos generators and not a full cryptosystem. Thus, we can say that those attempts did not succeed in proposing a lightweight cryptosystem that can be implemented in these devices.

In contrast to those presented above, this paper proposes a complete lightweight, and efficient cryptosystem with new confusion–diffusion layers. These operations are optimized to be implemented in different hardware boards with limited resources such as microcontrollers. In addition, the proposed solution makes no assumptions about data types. It is generic and open to any type of data (sensing value, text, voice, image, etc.). Moreover, the proposed solution goes through a complete security analysis (randomness, entropy, histogram, etc.), which validates the strength of the adopted method. Most existing papers used simulations to estimate the effectiveness of their algorithms. A real implementation of encryption algorithms on constrained IoT nodes provides more realistic and accurate results. Thus, we develop a lightweight scheme and then implement it on real constrained IoT nodes.

The main contributions of this paper are summarized as follows:

- New chaos-based cryptosystem to secure IoT communications has been developed,
- High-performance chaotic generator is designed using a discretized version of the Lorenz system. It can generate random keys that greatly improve the quality and security of the proposed cryptosystem,
- New chaotic permutations at the bit level are also proposed to improve the security of the cryptosystem,
- Significant statistical experiments are conducted to evaluate the effectiveness of the proposed cryptosystem. It includes key size analysis, randomness analysis and robustness against classical attacks,
- Validation of the proposed solution on the Mbed microcontroller NXP LPC1768 with low memory usage and fast encryption speed. These results confirm that the proposed cryptosystem is suitable for resource-constrained IoT nodes.

The remainder of this paper is organized as follows. Section 2 presents the background of IoT and chaos theory, as well as discussions of the related research literature. The proposed cryptosystem is described in Sect. 3. Section 4 validates the proposed cryptosystem with experimental results, both at implementation and statistical levels. Finally, Sect. 5 concludes this work and gives some future works.

2 Background

As mentioned in the previous section, this work aims to secure IoT communications based on a chaotic approach. Therefore, in this section, we will start by describing IoT concepts, then the chaotic systems.

2.1 IoT systems

By 2025, it is predicted that more than 75 billion connected devices will be in use [30]. This number corresponds to a 3-times increase of devices in 2019. These devices are not only smartphones and tablets but also devices that can perform various functions, such as data acquisition functions (sensors) as well as operations on an environment (actuators) [31]. The network of these devices defines the concept of the *Internet of Things* (IoT) [32–34].

The definition of IoT includes both conceptual and technical dimensions [35]. At a conceptual level, IoT refers to

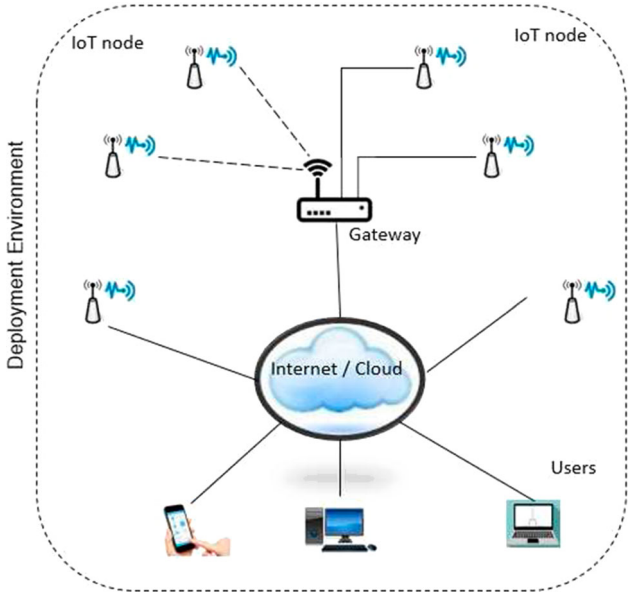

Fig. 1 Composition of an IoT network

connected physical devices that have their own identity and are able to communicate. At a technical level, it consists of a standardized digital identification (IP address, SMTP, HTTP protocols, etc.) of a physical device through a communication system that can be an RFID, Bluetooth, Wi-Fi, etc.

An IoT system is composed of IoT nodes that use communication media to transmit data to the cloud and processing stations to provide applications and services for end-users.

As shown in Fig. 1, IoT nodes collect and send their data through a gateway or directly (no gateway needed) to the cloud. IoT nodes can communicate with a gateway over wired or wireless communication links. Cloud can be used to store and process data. It can provide graphical interfaces to act on system settings.

IoT nodes are connected to the Internet; therefore, it is obvious that their data are targeted by various security threats requiring efficient encryption algorithms. Recently, significant research works have been reported on encryption techniques based on chaotic systems [36,37]. The development of a new robust security algorithm based on chaotic systems represents an alternative to overcome the limitations of conventional approaches [20].

2.2 Chaotic systems

In a linguistic context, chaos can be defined as a state of total confusion or general disorder. In a scientific context, chaos is related to unpredictability. A chaotic system is described as a dynamic behavior that appears random but is actually a disorder that is based on a deterministic system [38].

2.2.1 Characteristics of chaotic systems

In this work, the choice of a chaotic system is based on its characteristics such as its determinism, its random behavior, its sensitivity to initial conditions, and its unpredictability [38].

- *Deterministic*: A chaotic system is based on deterministic and non-probabilistic equations.
- *Aperiodic behavior*: A chaotic system remains irregular during its temporal evolution, it does not converge to any steady state.
- *Sensitivity to initial conditions*: For two arbitrary initial conditions very close at the input, the two corresponding trajectories at the output diverge exponentially.
- *Nonlinearity*: Chaotic systems are nonlinear systems since they are defined by nonlinear systems of equations.
- *Unpredictability*: It is extremely difficult to predict the evolution of a chaotic system, despite the fact that it is described by deterministic equations.

2.2.2 Types of chaotic systems

According to the equation that defines a chaotic system, we distinguish two types: continuous and discrete chaotic systems [39].

Continuous chaotic systems are described by a system of first order differential equations as follows:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = F(x(t), t) \tag{1}$$

where t represents time, $x(t)$ corresponds to the state vector of dimension $n \geq 3$, and F denotes the nonlinear dynamics of the system. Among chaotic systems with continuous time, we can mention the chaotic systems of Lorenz, Rossler, Chen, and Lu [40–42].

Discrete chaotic systems are generated by a temporal discontinuous progression characterized by a nonlinear recurrence equation. They are driven by an equation written in the form:

$$x(n+1) = G(x(n), n) \tag{2}$$

where n represents the index of the iteration, $x(n)$ corresponds to the state vector, and G is the nonlinear dynamics of the system. Examples of discrete-time chaotic systems are Henon Map, Logistic Map, Tent Map, and Chirikov Map [43–45].

Given the benefits of chaos theory, many researchers have proposed cryptosystems based on chaotic systems. In the following section, the details of the new cryptosystem are described.

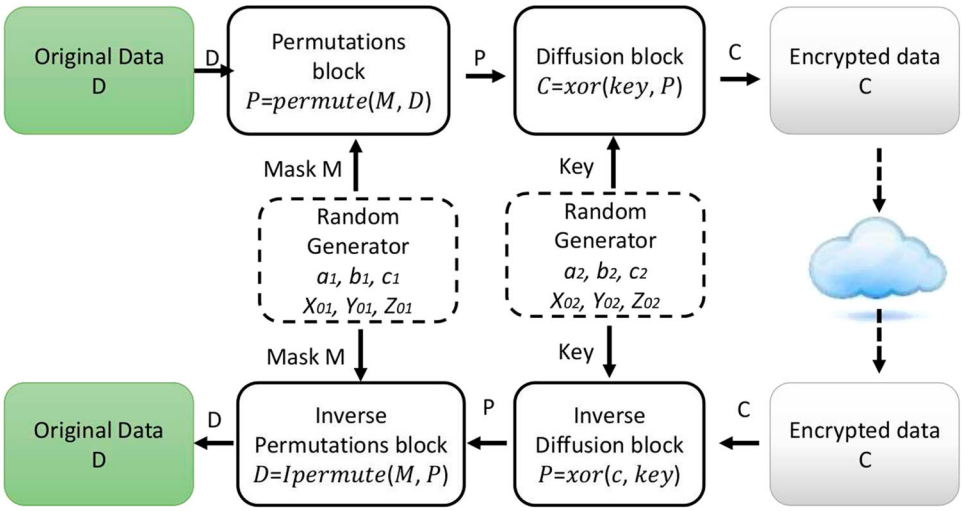

Fig. 2 An overview of the proposed chaos-based cryptosystem to secure IoT communications

3 Proposed cryptosystem

Handle critical data requires the development of cryptosystems that take into account the limited resources of certain hardware platforms in terms of memory, energy, and computing power.

3.1 Overview of the cryptosystem

As we pointed out in the previous section, the currently proposed approaches do not meet all the needs when designing a secure IoT network. Therefore, the main objective is to:

- Implement a powerful and less complex cryptosystem to secure IoT communications,
- Validate the proposed cryptosystem in real use case covering the whole IoT lifecycle from data acquisition to data exploitation.

As illustrated in Fig. 2, the proposed cryptosystem is based on three essential blocks: (1) *random generator block* to generate a key with high statistical proprieties, (2) *chaotic permutations block* to permute data with variable permutations, and (3) *diffusion block* to xor the output of the previous step with a key to produce ciphered data. In the following sections, we will discuss the details of each block.

3.2 Random generator

For the random generator, we have chosen a chaotic system. As illustrated in the second section, several systems can be used. Among them, *the Lorenz system* is adopted to generate random keys. This choice is based on its better statistical performances and low computational resource requirements compared to the existing chaotic systems [20]. Lorenz system is a deterministic chaotic system described by Eq. 3:

$$\begin{aligned} \frac{\mathrm{d}x(t)}{\mathrm{d}t} &= a(y-x) \\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} &= cx - y - xz \\ \frac{\mathrm{d}z(t)}{\mathrm{d}t} &= xy - bz \end{aligned} \tag{3}$$

where a, b, and c are the system parameters and x_0, y_0, and z_0 are the initial conditions. By knowing the initial conditions, it is possible to determine the system's evolution over time. However, with a slight difference in these values, the predictions will be fundamentally changed.

For such a system, the chaotic effect is in x, y, and z solutions. To solve Eq. 3, numerical solution methods can be used, such as the Runge–Kutta method. Figure 3 shows the simulation results of the Lorenz discrete chaotic system. These results represent the chaotic signals x, y, and z. These simulations are performed using the Runge–Kutta method under the MATLAB tool. This generator is used in the encryption and decryption processes.

3.3 Encryption process

In symmetric key algorithms, two cryptography techniques are used: permutation and diffusion. They obscure the relationship between the original data and the encrypted data [46].

3.3.1 Permutations block

To enhance security level, permutation is considered. Several permutation patterns exist in the literature. There are permutations of columns and rows in images, permutations of letters in text data, rotations in arrays, etc. [47].

In this work, we propose to achieve a new pseudo-random permutation of data bits using the permute function as follows:

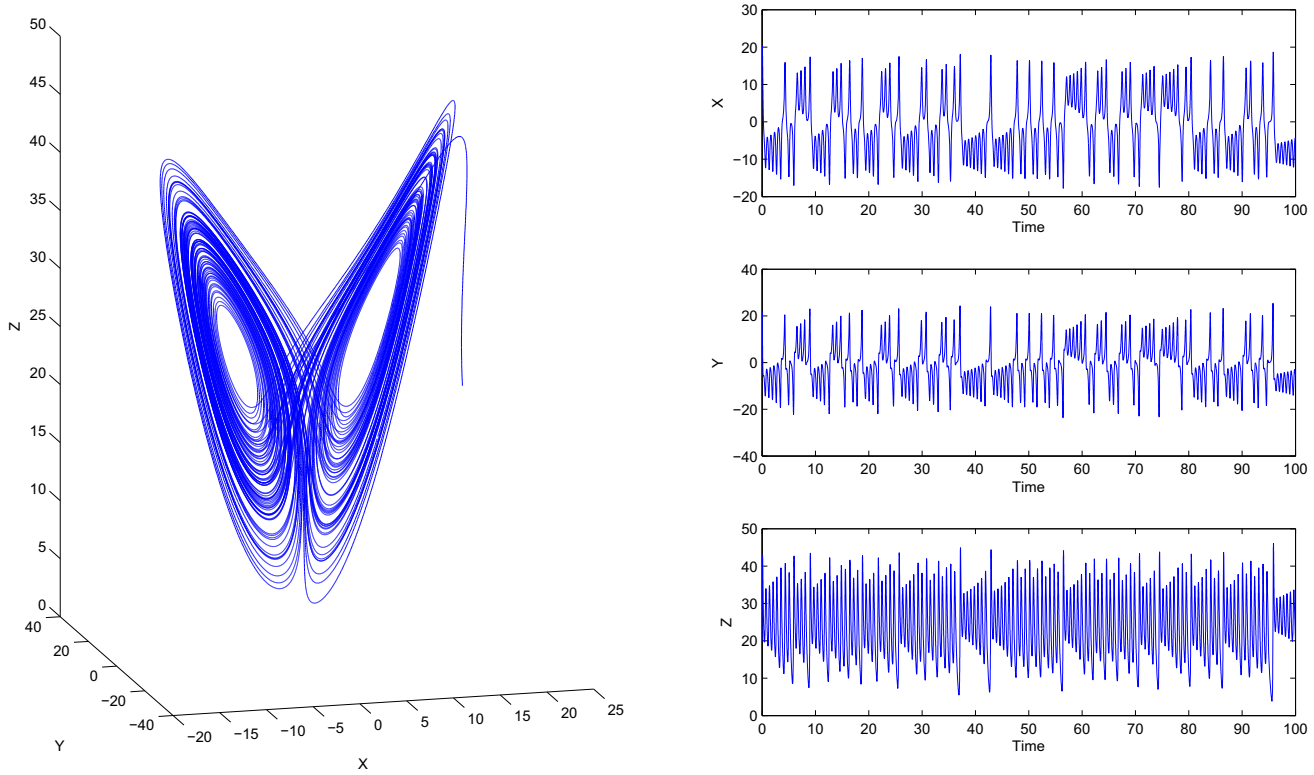

Fig. 3 Lorenz chaotic system: x, y, and z chaotic signals

$$P = permute(M, D) \tag{4}$$

where D represents the data, M represents the permutation mask generated by a chaotic generator, and P represents the permuted data. The permute function takes n bits of M and n bits of D to produce n bits of P. First, the permutation function examines the mask and data bit by bit. Each bit of the mask M is checked. If this bit is 0, the corresponding bit of P is set to the bit of D in the increasing order of D. Otherwise, the bit of P is set to the bit of D in the decreasing order. As shown in Fig. 4, we take a 32 bits data and a 32 bits mask and we get a 32 bits output. This process is also summarized in the pseudo-code of Algorithm 1.

Algorithm 1 Pseudo-code of the proposed permute function

Input: Data D (n bits), Mask M (n bits)
Output: Permuted data P (n bits)
Initialization: i=1, j=n
  for `each bit k of M` **do**
    if M_k = 0 **then**
      P_k = D_i
      i = i+1
    else
      $P_k = D_j$
      j = j-1
  end if
end for

3.3.2 Diffusion block

In this block, a diffusion process that spreads the redundancy of the permuted data over the cipher is proposed. In this step, the output of the permutation block is mixed with a key generated by the chaotic generator using the *XOR* operation:

$$C = xor(key, P) \tag{5}$$

In fact, each bit from the previous operation is XORed with one bit of the key. This process is repeated until the whole data is encrypted.

3.4 Decryption process

In symmetric encryption algorithms, the original data can be obtained from the encrypted data by applying the same tasks of the encryption step in reverse order. The encryption step starts with the permutation, then the diffusion. Therefore, the decryption step begins with the inverse diffusion and then the inverse permutation.

3.4.1 Inverse diffusion

This process obviously requires the generation of the same key used in encryption. The synchronization of the different

configurations (parameters and initial conditions) is a mandatory condition to decrypt the data correctly.

Using the chaotic generator, the random key is generated and used in the inverse diffusion. The permuted data is retrieved using the xor function:

$$P = xor(C, key) \tag{6}$$

Each bit of the encrypted data C is XORed with the corresponding bit of the Key. This process is repeated until the entire data is decrypted.

3.4.2 Inverse permutation

The inverse permutation block takes the permuted data P and the mask M to produce the data D. The function $Ipermute$ takes n bits of M and n bits of P to produce n bits of D as follows:

$$D = Ipermute(M, P) \tag{7}$$

Figure 5 illustrates this process with the permuted data P, the mask M, and the data D are coded on 32 bit.

Each bit of the mask M is checked. If this bit is 0, the corresponding bit in P is assigned to the bit of D in the increasing direction. Otherwise, the corresponding bit in P is assigned to the bit of D in the decreasing sense of P. This process is also summarized in the pseudo-code of Algorithm 2.

Algorithm 2 Pseudocode of the proposed inverse permute function

Input: Permuted data P (n bits), Mask M (n bits)
Output: Data D (n bits)
Initialization: i=1, j=n
  for each bit k of M **do**
    if M_k = 0 **then**
      $D_i = P_k$
      i = i+1
    else
      $D_j = P_k$
      j = j-1
  end if
end for

4 Performance evaluation

This section presents a use case, implementation results, and security analysis of the proposed solution.

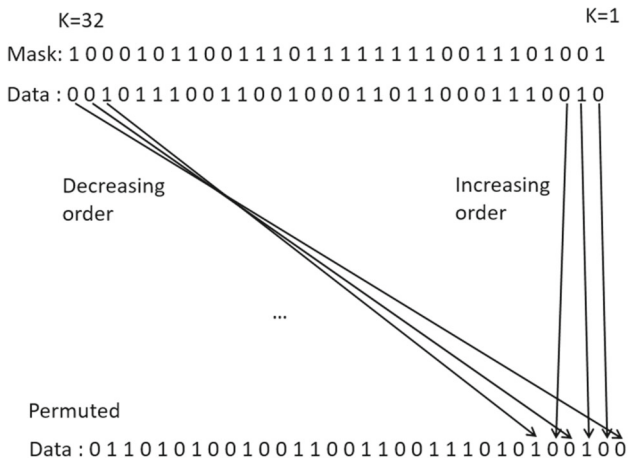

Fig. 4 Proposed permutation block

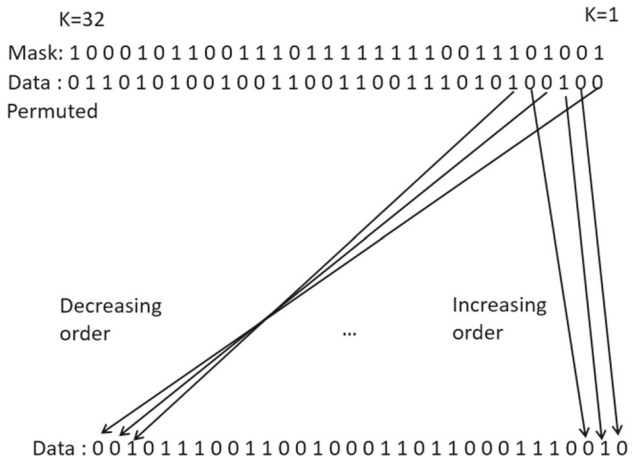

Fig. 5 Inverse process of the proposed permutation block

4.1 Use case and setup

To validate the effectiveness of the proposed solution, a popular scenario is used as a case study involving the monitoring of buildings. To achieve this goal, an IoT network is designed, composed of IoT nodes, gateways, and databases. In this network, IoT nodes send data through gateways that then transmit it to databases. In this solution, three levels are distinguished: IoT node level, gateway level, and storage level (see Fig. 6).

4.1.1 IoT node level

In this case study, IoT nodes are based on the microcontroller Mbed NXP LPC1768 as shown in Fig. 7. IoT nodes start by acquiring data with a configurable time interval. They can detect and measure different types of data. According to requirements, it is possible to associate these devices with cameras to send images/video. IoT nodes are programmed to encrypt data using the proposed cryptosystem. Next, they

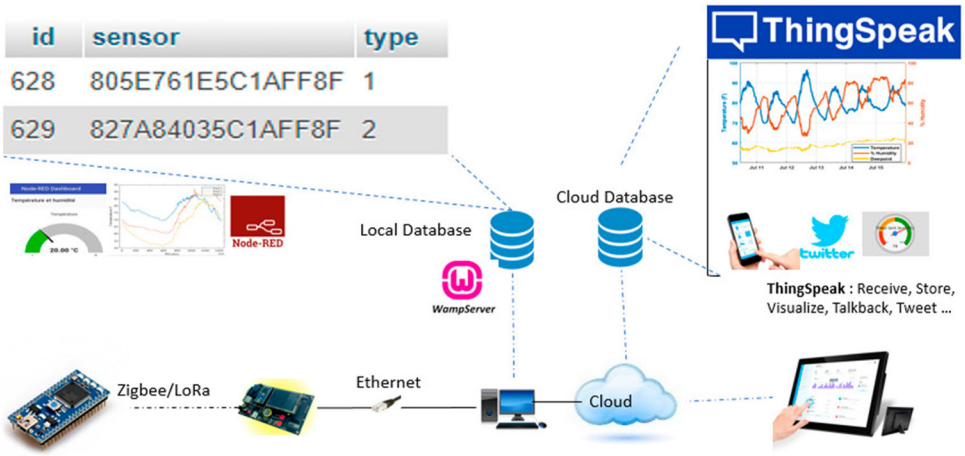

Fig. 6 An overview of the use case

send the encrypted data to the cloud directly or through the gateway.

4.1.2 Gateway level

The gateway acts as a relay between IoT nodes and the cloud. For communication links between IoT nodes and gateway, we can use different communication technologies such as LoRa and ZigBee. This choice depends on many parameters, such as distance and required data rate.

For our gateway, we also use the Mbed board (with a testbed for Mbed). It can receive data from IoT nodes through ZigBee technology, and then it sends the received data to databases with an Ethernet link. Figure 7 shows the gateway and five IoT nodes used in this use case.

4.1.3 Database level

To store and process data, we use two types of databases: local and cloud databases.

Local database stores data locally in the building. It is managed and hosted in the WampServer tool. This server provides two-stream Ethernet connections: one stream carrying the data sent by IoT nodes and another stream for data exploitation. We chose the Node-RED tool for the implementation of data exploitation [48]. Node-RED is a programming tool that connects hardware devices, APIs, and offers IoT services. It can run locally on a computer, Raspberry Pi, or through the cloud as offered by IBM Cloud, SenseTecnic FRED, Amazon Web Services, and Microsoft Azure. In Node-RED, a template is described to perform a decryption process using the proposed cryptosystem and then process data. Figure 6 shows an example of an interface that can notify users if one or more devices have not sent data after a configurable time interval. A remote e-mail notification service can also be developed.

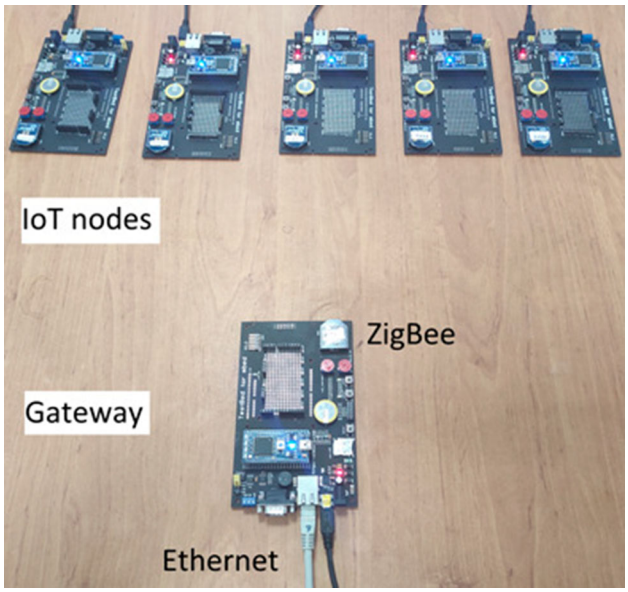

Fig. 7 The validation network composed of 5 IoT nodes and one gateway: Mbed NXP LPC1768

Cloud database represents existing IoT platforms such as ThingSpeak [49]. It is an open-source platform for data storage and exploitation using Internet Protocols. Regarding the use case, an account was created on ThingSpeak and its channels were configured to receive data. ThingSpeak offers public and private channels. The public channel is accessible to all users on the cloud via a url: https://thingspeak.com/channels/public.

After describing our use case (from the IoT node to the database), the results of the implementation of the cryptosystem will be presented.

4.2 Implementation of cryptosystem

The chaos-based cryptosystem is implemented on the Mbed microcontroller. The cryptosystem is based on Lorenz

Table 1 Memory usage for encryption and decryption steps in the Mbed microcontroller

Type	Size for encryption (kB)	Size for decryption (kB)	Max (kB)
Flash	45.4	50.2	512
RAM	8.8	12.7	32

chaotic generator. In the current implementation, the keys for permutation and xor blocks are x and y outputs of the generator. These two solutions are $32bit$ floats. Since the cryptosystem is built on binary permutation and xor operations, we have converted these data from float to binary using the IEEE754 standard. Finally, the keys are constructed using the least significant bits (LSBs) of the fractional part of x and y of the chaotic generator because of their rich dynamics. In the following, memory usage, encryption speed, and energy consumption are discussed.

4.2.1 Memory usage

To evaluate memory usage, Mbed IDE (https://ide.mbed.com/compiler/) is used. Table 1 summarizes the memory usage for the encryption and decryption processes implemented on the Mbed microcontroller.

Regarding Flash memory, the encryption program consists of 45.4 kilobytes, representing 8.86% of the 512 kilobytes of total memory. It is approximately the same amount for decryption steps (50.2 kilobytes, representing 9.80%).

Regarding RAM memory, the encryption program used 8.8 kilobytes. It is approximately 27% of a total of 32 kilobytes. The amount of RAM used for decryption steps is 12.7 kilobytes, representing 39.68%. With this performance, our cryptosystem is suitable for various hardware boards with limited memory resources.

Compared to existing works [27–29], the authors did not mention memory usage in their implementation. They provide only a description of the maximum memory supported by the experimentation board.

4.2.2 Cryptosystem speed

Cryptosystem execution speed is an essential characteristic of its deployment performances. It is particularly pertinent for real-time applications. To evaluate encryption and decryption speed, we took the abstract of this article as data to encrypt multitude of times. The obtained execution speeds were on average 79.4 KiB/s for encryption and 38.84 KiB/s for decryption.

Table 2 shows the results of the encryption speed of existing works. We notice that the encryption speed of our algorithm is higher than all the others, except for the work of Al-Haija et al. [17], which has an encryption speed of 190 kb/s. This difference is mainly due to the implemented RSA algorithm with a low key size (32 bits in this work) compared to our approach with the key size of 192 bits. With an encryption speed of 79.44kb/s, the proposed cryptosystem is fast enough for many types of IoT applications.

4.2.3 Energy consumption

IoT applications enable a wide range of services such as industrial control, home automation, agriculture, and e-health. In such applications, IoT nodes operate autonomously and have to survive for months and years with strict energy constraints. Therefore, the development of such applications requires knowing their own energy consumption.

In this work, the power consumption for encryption and decryption on IoT nodes is evaluated. The adopted approach is based on two steps. In the first step, the power consumption of standby activity (without encryption) is measured. In the second step, the power consumption with encryption activities is measured. The power consumption of the encryption process is equal to the difference between these two power measurements. This approach is simple and generic because it does not involve complex equipment for measuring the energy consumption at run-time.

Our results show that the power estimation for the encryption process is equal to $7\text{mA} \times 5\text{v} = 35\,\text{mw}$. As a result, the energy consumption of the encryption process is calculated using the power consumption and duration of this activity obtained during execution. Using the result of encryption speed (79.44Kbit/s), the average energy consumption is about 0.44 uJ/bit during the encryption process. This low value can be supported by the majority of constraints IoT nodes powered by batteries. Compared to recent works [27–29], the authors did not provide and discuss any results on energy consumption.

4.3 Security analysis

Tests and analysis are used to determine whether a cryptosystem meets the requirements or not. In this section, we have selected a number of tests that are commonly used in the literature.

4.3.1 NIST statistical test

NIST test suite is a statistical package that includes fifteen tests. It tests the randomness of binary or hexadecimal sequences produced by generators [51]. This test takes as input binary sequences of size at least 1 Megabit. It returns a P-value for each test; this value must be greater than 0.01 to assume that the test is passed or not [52].

The results of our generator are shown in Table 3. All *P-Value* is higher than the threshold (0.01). According to NIST,

Table 2 Encryption speed results

Type	Proposed scheme	Adnan et al. [50]	Al-Haija et al. [17]	Lee et al. [15]
Encryption speed (KiB/s)	79.44	76	190	0.29

Table 3 NIST test results

Type	*P* value	Pass/fail
Frequency Test	0.739918	Pass
Block Frequency Test	0.213309	Pass
Cumulative Sums Test Up	0.311542	Pass
Cumulative Sums Test Down	0.437274	Pass
Runs Test	0.911413	Pass
Long Runs	0.010606	Pass
Rank Test	0.311542	Pass
Discrete Fourier Transform Test	0.534146	Pass
Non-overlapping Template Matching Test	0.637119	Pass
Overlapping Template Matching Test	0.242986	Pass
Universal Statistical Test	0.371101	Pass
Approximate Entropy Test	0.739918	Pass
Random Excursions Test	0.350485	Pass
Random Excursions Variant Test	0.585209	Pass
Serial Test 1	0.052778	Pass
Serial Test 2	0.964295	Pass
Linear Complexity Test	0.484646	Pass

the randomness of the generated keys is validated. A recent work [27] did also the Nist test to validate key randomness. Other works [28,29] did not mention any randomness tests.

4.3.2 DIEHARD statistical test

DIEHARD test package was introduced in 1996 by George Marsaglia [53]. It consists of a series of fifteen statistical tests to evaluate whether a generator produces random numbers. The DIEHARD test returns a value p which should be evenly distributed over the range [0,1]. The bitstream fails the test when the p-values are 0 or 1 [53].

The results of the DIEHARD tests are presented in Table 4. We notice that all "P-Value" are different from 0 and 1. According to DIEHARD, this confirms the validation of the random behavior of the generated keys.

Table 4 DIEHARD test results

Type	*P* value	Pass/fail
Birthday spacings test	0.473781	Pass
Overlapping 5-permutations test	0.512234	Pass
Binary rank test for 31 × 31 matrices	0.322461	Pass
Binary rank test for 32 × 32 matrices	0.428371	Pass
Binary rank test for 6 × 8 matrices	0.700730	Pass
DNA test	0.791100	Pass
Count the 1's TEST	0.618574	Pass
Parking lot test	0.294111	Pass
Minimum distance test	0.977207	Pass
3D spheres test	0.683966	Pass
SQEEZE test	0.008152	Pass
Overlapping sums test	0.024550	Pass
Runs test up	0.867289	Pass
Runs test down	0.101047	Pass
Craps test	0.704480	Pass

4.3.3 Key size analysis

In this section, a key size analysis is performed to study the robustness of the key generator against classical brute force attacks. If a generator can produce more than 2^{100} different key combinations, then it is considered unbreakable against this attack [36].

In this work, the generated key depends on 6 values: three initial conditions (x_0, y_0, and z_0) and three parameters (a, b, and c). Each value is encoded on 32 bits. Therefore, the key space is $2^{32} \times 2^{32} \times 2^{32} \times 2^{32} \times 2^{32} \times 2^{32} = 2^{192} \geq 2^{100}$. This

value is far above the required keyspace making this scheme robust against brute force attacks. In addition, the proposed chaotic system offers an equal or larger keyspace compared to the literature [54]. Compared to the encryption algorithms proposed [27–29], the proposed schema has a larger key size.

4.3.4 Key sensitivity analysis

Chaotic systems are very sensitive to tiny changes in the initial conditions and parameters of the chaotic system. In this section, we analyze this feature [36]. As shown in Fig. 8, we took two keys generated by the generator with tiny changes in the parameters of both systems (all initial conditions and parameters are the same, except that $a = 10.0$ for the first and $a = 10.1$ for the second). Then, we use the first generated key in the cryptosystem to encrypt the original data and then we used the second key in decryption. We notice that despite the fact that they are generated with very similar parameters, the obtained decrypted text is totally different. This difference shows that the proposed encryption solution is very sensitive to small changes in the initial conditions and parameters of the generated keys. It should be noted that the length of the encrypted data with our cryptosystem is the same as the length of the original data. Therefore, there is no extra cost to communication.

This feature is considered in the work [27]. However, other works have not provided any results regarding this metric despite its importance, especially in the case where IoT nodes send large amounts of data.

4.3.5 Differential analysis

Differential cryptanalysis studies the impact of the variation of input data on their encrypted data outputs [55]. The encryption algorithm must be resistant to differential attacks, thus a high sensitivity to input information such as the secret keys of the encryption system. There are two metrics that characterize the resistance to this type of attack, which is the Number of Pixels Change Rate (NPCR) and the Unified Average Changing Intensity (UACI). They measure the sensitivity of the cipher text to the key used in the encryption process.

Assuming that there are two very close keys: key_1 and key_2. Using the encryption algorithm, two ciphertexts C_1 and C_2 are generated regarding key_1 and key_2, respectively.

The NPCR is described by Eq. 8:

$$\text{NPCR} = \frac{\sum_{i=1}^{N} D_i}{N} \times 100\% \tag{8}$$

where

$$D_i = \begin{cases} 0, & \text{if } C_{1i} = C_{2i} \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

The UACI is described by Eq. 10:

$$\text{UACI} = \frac{\sum_{i=1}^{N} |C1_i - C2_i|}{255 * N} \times 100\% \tag{10}$$

We encrypted data using two keys with tiny differences in their values. Then, we evaluated UACI and NPCR values corresponding to two encrypted data. Table 6 shows the UACI/NPCR of our schema compared to the existing approach. The results show that our encryption algorithm has an NPCR value of 99.63 % which is greater than the one proposed by Das et al. [56] but less than the algorithm of Guerrero et al. [27].

Regarding UACI results, our schema outperforms all the other approaches. So, it can resist differential attacks.

4.3.6 Entropy analysis

Any cryptosystem is susceptible to entropy attack if the symbols in the encrypted data do not have the same probability of occurrence (uniform distribution). To measure the randomness of the encrypted data, the Shannon entropy H can be used as follows:

$$H = -\sum_{i} P_i \times \text{Log}_2(P_i) \tag{11}$$

where P_i is the probability of occurrence of a given symbol i. For 2^N possible symbols, the maximum entropy is N. In the case of text data, the total number of symbols is 2^8 symbols, so the optimal value of the entropy H should be 8. We calculated the entropy of the plaintext and the corresponding ciphertext. Table 7 shows the obtained results. The entropy of the ciphertext generated by our cryptosystem is 7.99, which is very close to the maximum theoretical value of 8.

Compared to existing work, our algorithm has the same entropy value of 7.99 as Guerrero et al. [27] and more than the algorithm of Das et al. [56]. Regarding this value, the proposed cryptosystem can generate a ciphertext that includes all the considered symbols with almost the same probability, achieving a uniform distribution. In other words, the proposed scheme is robust against entropy attacks.

4.3.7 Statistical histograms

To investigate the stability against statistical attacks, we analyze different histograms of the original data and the corresponding encrypted data. We take the abstract of this article

Table 5 Key size results

Type	Proposed scheme	Guerrero et al. [27]	Yabu et al. [28]	Ragab et al. [29]
Key (bits)	192	128	128	128

Fig. 8 Encrypted and decrypted data extracted from the Mbed microcontroller with a tiny difference in keys

```
--------------------Original data-------------------------
"city Smart is a recent concept that is adopted by many
researchers and authorities around the world."

--------------------Encrypted data-------------------------
™¨{ "°ópB \Ò?¤êNm'`DÅ:▯dÄ
Æ'K)¥¸‹*šÙ( ▯ãwýê;ªï ¯1™_▯q²ÄŠ ,÷yèk„ëãRUF@¬»▯µ³r6×ƒÛ ?
Œ™æ°ÁÄ)ý.%üõx cNs

---------Data decrypted with different parameters --------
;™> `A›F▯ ['ÝÒ1Ýáó!dfS‡¬▯—LŠ³▯ÿZï)úm «¼6¥▯▯¦âDjKpG…(n§▯ I !
  Ñ ë<dV▯œ€½¤°È ž▯ ¸{±^sÎ'/û€▯Ë S èÉÜ qv% %▯—
```

Table 6 NPCR and UACI results

Type	Proposed schema	Guerrero et al. [27]	Das et al. [56]
NPCR (%)	99.63	99.80	99.59
UACI (%)	33.50	33.48	32.99

as the data to be encrypted. Each byte is composed of 8 bits representing a number between 0 and 255.

Figure 9 shows the histogram's original data and the encrypted data. Unlike the original data, the encrypted data have a uniform distribution. This demonstrates that the proposed algorithm is resistant to statistical attacks.

This characteristic is also investigated in [27,56], they compare the histogram of the original image and the encrypted one. The encrypted image has a uniform distribution similar to our results.

4.3.8 Comparison with the closest related works

Currently, there are a few implementations of the chaos-based encryption algorithm for IoT nodes with limited resources. To study these implementations, we considered comparison criteria regarding the chaotic map, implementation boards, data type (specific or generic), memory costs, encryption speed, keyspace, and security analysis. Table 8 presents the comparison results of the most recent approaches dealing with lightweight cryptosystems.

In contrast to those presented above, the proposed solution makes no assumptions about data types. It is generic and open to any type of data (sensing value, text, voice, image, etc.). Moreover, our implementation is supported by a complete security analysis, which validates the application of our method. We also measured some important implementation values such as low memory usage, fast encryption speed, and low energy consumption of our encryption algorithms. The existing works did not mention their implementation results. In addition, we have proposed a complete cryptosystem with new confusion and diffusion tasks. Both operations can be further optimized to be implemented in different hardware boards. Regarding flexibility, our algorithm is able to support minor modifications to meet the needs of more users such as changing the number of bits in confusion and diffusion process.

5 Conclusion

To secure IoT communications, this paper develops a new lightweight and efficient chaos-based cryptosystem for resource-constrained IoT nodes. The proposed cryptosystem is based on three essential components: *a chaotic generator* based on the Lorenz system to generate random keys, a

Table 7 Entropy results using the proposed cryptosystem in the Mbed microcontroller

Type	Plaintext entropy	Ciphertext entropy	Maximum value
Proposed cryptosystem	4.24	7.99	8
Guerrero et al. [27]	–	7.99	8
Das et al. [56]	7.52	7.97	8

Table 8 Comparison of the proposed solution with the closest related works

Type	Proposed scheme	Guerrero et al. [27]	Yabu et al. [28]	Ragab et al. [29]	Das et al. [56]	Adnan et al. [50]	Al-Haija et al. [17]	Lee et al. [15]
Proposed schema	Lorenz map	Hénon map	SAEAES	HYBRID	logistic map	AABET	RSA	AES
Confusion algorithm	✓	–	✓	–	–	–	–	✓
Diffusion algorithm	✓	✓	✓	✓	✓	✓	✓	✓
Keys space (bits)	192	128	128	128	–	6144	32	128
key sensitivity	✓	–	–	–	–	–	–	–
Randomness	✓	✓	–	–	–	–	–	–
Histogram	✓	✓	–	–	✓	–	–	–
Entropy	7.99	7.99	–	–	7.97	–	–	–
NPCR (%)	99.63	99.80	–	–	99.59	–	–	–
UAC (%)	33.50	33.48	–	–	32.99	–	–	–
Hardware board	Mbed LPC 1768	PIC 16F877A	ARM M0	Atmel AVR	Atmega32	ARM M7	Arduino Mega	Atmega
Data type	Generic	Image	Text	Text	Image	Text	Text	Sinsing data
Memory cost (KiB)	45	–	–	–	–	–	–	–
Encryption speed (KiB/s)	79.44	–	–	–	–	76	190	0.29
Energy consumption (uJ/bit)	0.44	–	–	–	–	–	–	–
Flexibility	✓	–	–	–	–	–	–	–

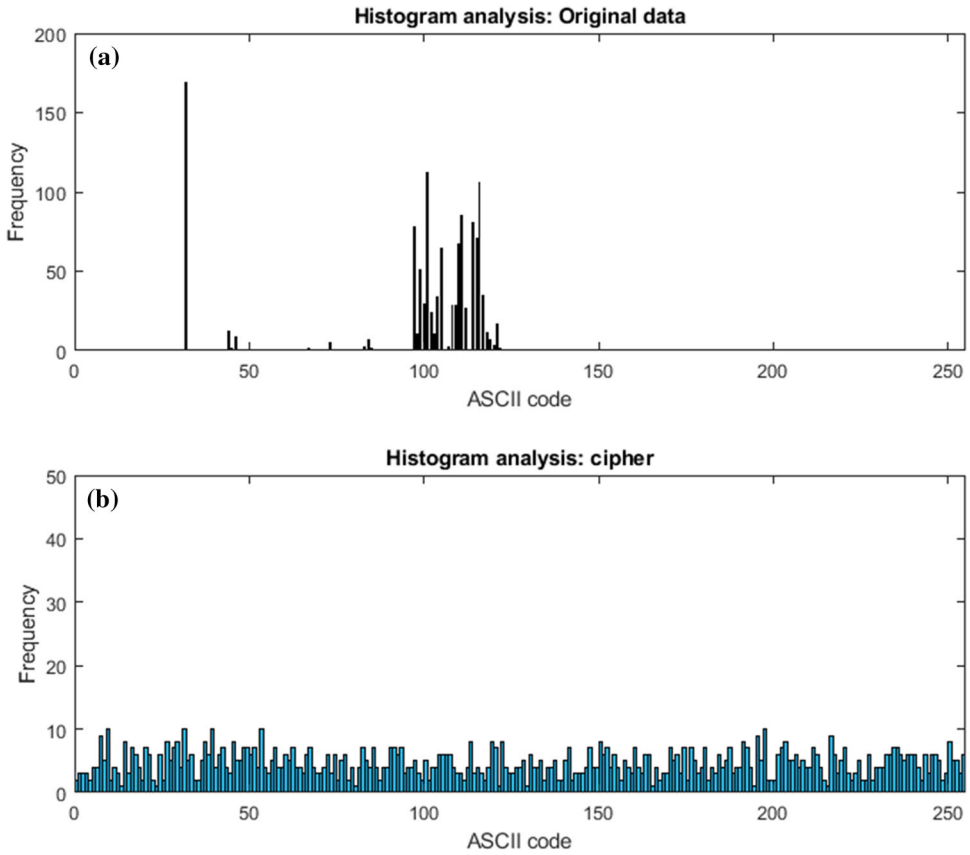

Fig. 9 Histogram analysis: **a** original data **b** cipher

new chaotic permutation that creates confusion, and *diffusion* using the xor operation with a chaotic key. A real use case is used to investigate the effectiveness and suitability of the proposed solution for resource-constrained IoT nodes. Through the implementation results, we have demonstrated that the proposed cryptosystem is suitable for Mbed microcontroller with low memory usage, fast encryption/decryption speed, and low energy consumption. In addition, statistical results such as the NIST tests, key size, key sensitivity, entropy, differential analysis, and statistical histogram analysis show that this cryptosystem is very promising to provide a robust tool against many attacks.

In future work, we intend to improve the proposed cryptosystem by proposing a key sharing mechanism. A lightweight security protocol that involves authentication of deployed IoT devices and hash functions to ensure data integrity are also in the perspective of this work.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Cosgrave, J.: Ready to respond:skills gaps for responding to humanitarian crises in urban settings in the wash and shelter sectors (2013). https://www.urban-response.org/help-library. Accessed Feb 2022
2. UD. of Transportation: Smart city challenge (2021). https://www.transportation.gov/smartcity. Accessed Oct 2021
3. Yaacoub, J.-P.A., Noura, H.N., Salman, O., Chehab, A.: Robotics cyber security: vulnerabilities, attacks, countermeasures, and recommendations. Int. J. Inf. Secur. **66**, 1–44 (2021). https://doi.org/10.1007/s10207-021-00545-8
4. Stellios, I., Kotzanikolaou, P., Psarakis, M., Alcaraz, C., Lopez, J.: A survey of iot-enabled cyberattacks: assessing attack paths to critical infrastructures and services. IEEE Commun. Surv. Tutor. **20**(4), 3453–3495 (2018). https://doi.org/10.1109/COMST.2018.2855563
5. Rajbir, K., Navroop, K., Sood Sandeep, K.: Security in iot network based on stochastic game net model [j]. Int. J. Netw. Manag. **27**(4), 1–13 (2017). https://doi.org/10.1002/nem.1975
6. Sakiz, F., Sen, S.: A survey of attacks and detection mechanisms on intelligent transportation systems: Vanets and iov. Ad Hoc Netw. **61**, 33–50 (2017). https://doi.org/10.1016/j.adhoc.2017.03.006
7. Yang, Q., Yang, J., Yu, W., An, D., Zhang, N., Zhao, W.: On false data-injection attacks against power system state estimation: modeling and countermeasures. IEEE Trans. Parallel Distrib. Syst. **25**(3), 717–729 (2013). https://doi.org/10.1109/TPDS.2013.92
8. Obaidat, M.S., Rana, S.P., Maitra, T., Giri, D., Dutta, S.: Biometric security and internet of things (iot). In: Biometric-Based Physical and Cybersecurity Systems, pp. 477–509. Springer (2019). https://doi.org/10.1007/978-3-319-98734-7_19

9. Prevezianou, M.F.: Wannacry as a creeping crisis. Underst. Creep. Cris. **37**, 2021 (2021). https://doi.org/10.1007/978-3-030-70692-0_3
10. Farion-Melnyk, A., Rozheliuk, V., Slipchenko, T., Banakh, S., Farion, M., Bilan, O.: Ransomware attacks: risks, protection and prevention measures. In: 2021 11th International Conference on Advanced Computer Information Technologies (ACIT), pp. 473–478. IEEE (2021). https://doi.org/10.1109/ACIT52158.2021.9548507
11. Fernandes, D.A., Soares, L.F., Gomes, J.V., Freire, M.M., Inácio, P.R.: Security issues in cloud environments: a survey. Int. J. Inf. Secur. **13**(2), 113–170 (2014). https://doi.org/10.1007/s10207-013-0208-7
12. Singh, S., Sharma, P.K., Moon, S.Y., Park, J.H.: Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions. J. Ambient Intell. Hum. Comput. **66**, 1–18 (2017). https://doi.org/10.1007/s12652-017-0494-4
13. Seyhan, K., Nguyen, T.N., Akleylek, S., Cengiz, K.: Lattice-based cryptosystems for the security of resource-constrained iot devices in post-quantum world: a survey. Clust. Comput. **66**, 1–20 (2021). https://doi.org/10.1007/s10586-021-03380-7
14. McKay, K., Bassham, L., Turan, M., Mouha, N.: Report on lightweight cryptography. National Institute of Standards and Technology internal report 8114 (2017). https://doi.org/10.6028/NIST.IR.8114
15. Lee, H., Lee, K., Shin, Y.: Aes implementation and performance evaluation on 8-bit microcontrollers, arXiv preprint arXiv:0911.0482 (2009). https://arxiv.org/abs/0911.0482
16. Omrani, T., Rhouma, R., Becheikh, R.: Licid: a lightweight image cryptosystem for iot devices. Cryptologia **43**(4), 313–343 (2019). https://doi.org/10.1080/01611194.2018.1563009
17. Al-Haija, Q.A., Al Tarayrah, M., Al-Qadeeb, H., Al-Lwaimi, A.: A tiny rsacryptosystem based on arduino microcontroller useful for small scalenetworks. Procedia Comput. Sci. **34**, 639–646 (2014)
18. Aishwarya, R., Sreerangaraju, M.: Enhanced security using dna cryptography. Int. Res. J. Eng. Technol. **6**, 3193–3196 (2019)
19. Pasupuleti, S.K., Varma, D.: Lightweight ciphertext-policy attribute-based encryption scheme for data privacy and security in cloud-assisted iot. In: Real-Time Data Analytics for Large Scale Sensor Data, pp. 97–114. Elsevier (2020)
20. Chen, S., Yu, S., Lü, J., Chen, G., He, J.: Design and fpga-based realization of a chaotic secure video communication system. IEEE Trans. Circuits Syst. Video Technol. **28**(9), 2359–2371 (2017). https://doi.org/10.1109/TCSVT.2017.2703946
21. Nguyen, N., Pham-Nguyen, L., Nguyen, M.B., Kaddoum, G.: A low power circuit design for chaos-key based data encryption. IEEE Access **8**, 104432–104444 (2020)
22. Nesa, N., Banerjee, I.: A lightweight security protocol for iot using Merkle hash tree and chaotic cryptography. In: Advanced Computing and Systems for Security, pp. 3–16. Springer (2020). https://doi.org/10.1007/978-981-13-8969-6_1
23. Ahmad, J., Larijani, H., Emmanuel, R., Mannion, M., Qureshi, A.-U.-H.: Secure occupancy monitoring system for iot using lightweight intertwining logistic map. In: 2018 10th Computer Science and Electronic Engineering (CEEC), pp. 208–213 (2018). https://doi.org/10.1109/CEEC.2018.8674208
24. Akgül, A., Kaçar, S., Arıcıoğlu, B., Pehlivan, I.: Text encryption by using one-dimensional chaos generators and nonlinear equations. In: 2013 8th International Conference on Electrical and Electronics Engineering (ELECO), pp. 320–323. IEEE (2013). https://doi.org/10.1109/ELECO.2013.6713853
25. Rajendran, S., Doraipandian, M.: Chaos based secure medical image transmission model for IoT-powered healthcare systems. IOP Conf. Ser. Mater. Sci. Eng. **1022**, 012106 (2021). https://doi.org/10.1088/1757-899X/1022/1/012106
26. Azzaz, M., Krimil, M.: A new chaos-based text encryption to secure gps data. In: 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT), pp. 294–299. IEEE (2018). https://doi.org/10.1109/SaCoNeT.2018.8585703
27. García-Guerrero, E., Inzunza-González, E., López-Bonilla, O., Cárdenas-Valdez, J., Tlelo-Cuautle, E.: Randomness improvement of chaotic maps for image encryption in a wireless communication scheme using pic-microcontroller via zigbee channels. Chaos Solitons Fract. **133**, 109646(2020)
28. Yabu, M., Sakiyama, K., Sugawara, T.: Low-memory implementation of authenticated encryption algorithm saeaes on arm cortex-m0 microcontroller. In: 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE), pp. 181–185. IEEE (2020)
29. Ragab, A., Selim, G., Wahdan, A., Madani, A.: Robust hybrid lightweight cryptosystem for protecting iot smart devices. In: International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, pp. 5–19. Springer (2019). https://doi.org/10.1007/978-3-030-24900-7_1
30. Company, S.: Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (2021). https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. Accessed Oct 2021
31. Kifouche, A., Hamouche, R., Kocik, R., Rachedi, A., Baudoin, G.: Model driven framework to enhance sensor network design cycle. Trans. Emerg. Telecommun. Technol. **30**(8), e3560 (2019). https://doi.org/10.1002/ett.3560
32. Lv, Z., Qiao, L., Kumar Singh, A., Wang, Q.: Ai-empowered iot security for smart cities. ACM Trans. Internet Technol. **21**(4), 1–21 (2021). https://doi.org/10.1145/3406115
33. Rachedi, A., Rehmani, M.H., Cherkaoui, S., Rodrigues, J.J.: IEEE access special section editorial: the plethora of research in internet of things (iot). IEEE Access **4**, 9575–9579 (2016). https://doi.org/10.1109/ACCESS.2016.2647499
34. Stergiou, C., Psannis, K.E.: Recent advances delivered by mobile cloud computing and internet of things for big data applications: a survey. Int. J. Netw. Manag. **27**(3), e1930 (2017). https://doi.org/10.1002/nem.1930
35. I.I. Initiative, et al.: Towards a definition of the internet of things (iot) (2015)
36. Alvarez, G., Li, S.: Some basic cryptographic requirements for chaos-based cryptosystems. Int. J. Bifurc. Chaos **16**(08), 2129–2151 (2006). https://doi.org/10.1142/S0218127406015970
37. Li, J., Zhang, W., Kumari, S., Choo, K.-K.R., Hogrefe, D.: Security analysis and improvement of a mutual authentication and key agreement solution for wireless sensor networks using chaotic maps. Trans. Emerg. Telecommun. Technol. **29**(6), e3295 (2018). https://doi.org/10.1002/ett.3295
38. Lorenz, E.N.: Deterministic nonperiodic flow. J. Atmos. Sci. **20**(2), 130–141 (1963). https://doi.org/10.1175/1520-0469(1963)0200130:DNF2.0.CO;2
39. Sparrow, C.: The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors. Springer, Berlin (2012)
40. Moon, S., Baik, J.-J., Seo, J.M.: Chaos synchronization in generalized Lorenz systems and an application to image encryption. Commun. Nonlinear Sci. Numer. Simul. **96**, 105708 (2021). https://doi.org/10.1016/j.cnsns.2021.105708
41. Li, C., Chen, G.: Chaos in the fractional order Chen system and its control. Chaos Solitons Fract. **22**(3), 549–554 (2004). https://doi.org/10.1016/j.chaos.2004.02.035
42. Agiza, H.: Chaos synchronization of lü dynamical system. Nonlinear Anal. Theory Methods Appl. **58**(1–2), 11–20 (2004). https://doi.org/10.1016/j.na.2004.04.002
43. Hu, T.: Discrete chaos in fractional Hénon map. Appl. Math. **2014**, 66 (2014). https://doi.org/10.4236/am.2014.515218

44. Wu, G.-C., Baleanu, D.: Chaos synchronization of the discrete fractional logistic map. Signal Process. **102**, 96–99 (2014). https://doi.org/10.1016/j.sigpro.2014.02.022
45. Yang, T.: A survey of chaotic secure communication systems. Int. J. Comput. Cognit. **2**(2), 81–130 (2004)
46. Lee, R.B., Shi, Z., Yang, X.: Efficient permutation instructions for fast software cryptography. IEEE Micro **21**(6), 56–69 (2001). https://doi.org/10.1109/40.977759
47. Alfarano, G.N., Beierle, C., Isobe, T., Kölbl, S., Leander, G.: Shiftrows alternatives for aes-like ciphers and optimal cell permutations for midori and skinny. In: IACR Transactions on Symmetric Cryptology, pp. 20–47 (2018). https://tosc.iacr.org/index.php/ToSC/article/view/887
48. Lekić, M., Gardašević, G.: Iot sensor integration to node-red platform. In: 2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH), pp. 1–5. IEEE (2018). https://doi.org/10.1109/INFOTEH.2018.8345544
49. M. Inc: Understand your things: the open iot platform with Matlab analytics (2021). https://thingspeak.com/. Accessed Oct 2021
50. Adnan, S.F.S., Isa, M.A.M., Hashim, H.: Analysis of asymmetric encryption scheme, aa β performance on arm microcontroller. In: 2017 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), pp. 146–151. IEEE (2017)
51. Bassham, L.E., III, Rukhin, A.L., Soto, J., Nechvatal, J.R., Smid, M.E., Barker, E.B., Leigh, S.D., Levenson, M., Vangel, M., Banks, D.L., et al.: Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications (2010). https://doi.org/10.5555/2206233
52. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E.: A statistical test suite for random and pseudorandom number generators for cryptographic applications, tech. rep., Booz–Allen and Hamilton Inc Mclean, VA (2001)
53. Marsaglia, G.: Diehard: a battery of tests of randomness. http://stat.fsu.edu/geo (1996)
54. Bouteghrine, B., Tanougast, C., Sadoudi, S.: Novel image encryption algorithm based on new 3-d chaos map. Multimedia Tools Appl. **66**, 1–23 (2021). https://doi.org/10.1007/s11042-021-10773-8
55. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991). https://doi.org/10.1007/BF00630563
56. Das, A., Hajra, S., Mandal, M.: Rgb image encryption using microcontroller atmega 32. Microsyst. Technol. **27**(2), 409–417 (2021). https://doi.org/10.1007/s00542-018-3980-5