ISSN: 0067-2904

A lightweight AES Algorithm Implementation for Secure IoT Environment

Meryam Saad Fadhil *, Alaa Kadhim Farhan, Mohammad Natiq Fadhil
Computer Sciences Department, University of Technology, Baghdad, Iraq

Received: 2/9/2020 Accepted: 28/10/2020

Abstract

In recent years, the rapid development in the field of wireless technologies led to the appearance of a new topic, known as the Internet of things (IoT). The IoT applications can be found in various fields of our life, such as smart home, health care, smart building, and etc. In all these applications, the data collected from the real world are transmitted through the Internet; therefore, these data have become a target of many attacks and hackers. Hence, a secure communication must be provided to protect the transmitted data from unauthorized access. This paper focuses on designing a secure IoT system to protect the sensing data. In this system, the security is provided by the use of Lightweight AES encryption algorithm to encrypt the data received from physical environment. The hardware used in this proposal is the Raspberry Pi 3 model B and two types of sensors. The LAES algorithm was embedded inside the Raspberry in order to protect the sensing data, that come from sensors connected to the Raspberry Pi, before sending them through the network. The analysis results show that the proposed IoT security system consumes less time in encryption/decryption and has high throughput when compared with others from related work. Its throughput is higher in about 19.24% than the value reported for one system in the related studies.

Keywords: IoT systems, LAES, Raspberry Pi, Sensors, Lightweight cryptography

تطبيق خوارزمية AES المخففة لحماية بيئة انترنت الاشياء

مريم سعد فاضل*, علاء كاظم فرحان, محمد ناطق فاضل
قسم علوم الحاسوب, الجامعة التكنولوجية, بغداد, العراق

الخلاصة

في السنوات الأخيرة ، التطور السريع الذي حصل في مجال التكنولوجيات اللاسلكية أدى إلى الظهور بموضوع جديد هو إنترنت الأشياء (IoT). حيث تطبيقات إنترنت الأشياء اصبحت موجودة في مجالات الحياة المختلفة مثل المنزل الذكي, الرعاية الصحية, البناء الذكي وغيرها. في جميع هذه التطبيقات, البيانات التي يتم جمعها من العالم الحقيقي سيتم ارسالها خلال شبكة الإنترنت، وبالتالي أصبحت هدفا للعديد من الهجمات والمتسللين. لذلك توفير اتصال آمن لحماية هذه البيانات من الوصول غير المصرح به اصبحت مسألة جدا مهمة. تركز هذه الورقة على تصميم نظام انترنت الاشياء الامن لحماية بيانات الاستشعار. حيث توفير الأمان يتم من خلال استخدام خوارزمية التشفير AES خفيفة الوزن لتشفير البيانات المستلمة من البيئة الفعلية. الأجهزة المستخدمة في هذا المقترح هو Raspberry Pi 3 نموذج B ونوعين من أجهزة

*Email: 110897@student.uotechnology.edu.iq

الاستشعار. حيث تم تضمين خوارزمية LAES داخل Raspberry من أجل حماية بيانات أجهزة الاستشعار المتصلة بـ Raspberry Pi قبل إرسالها عبر الشبكة. تظهر نتائج التحليل أن نظام IoT الامن المقترح يستهلك وقتًا أقل في كل من عملية التشفير/فك التشفير وكذلك لديه إنتاجية عالية عند مقارنته بغيره من الأعمال السابقة. حيث الإنتاجية لهذا النظام أعلى بحوالي 19.24٪ من أحد الأنظمة في الدراسات السابقة.

1. Introduction

Recently, the rapid development in communications, computer science and wireless networking technology has given rise to the growth of a new topic, known as the Internet of Things (IoT) [1, 2]. The IoT allows a connection between every object in the real world with the virtual world. This connection can be provided using some technologies such as Wireless Sensor Networking (WSN), Radio Frequency Identification (RFID), or merging between these technologies, machine-to-machine interfacing (M2M), cloud servicing, etc. [1, 3, 4]. The IoT can be found in various applications of our daily life such as smart home, health care, smart city, smart farming, smart factories and so on [5]. With the IoT systems, all the data collected from the real world are transmitted through the Internet; therefore, these data have become a target of cyber-attacks and hackers [6]. Hence, a secure communication must be provided to protect these transmitted data [7].

To provide security to IoT systems, such as confidentiality, integrity and authentication, the solution is to use an appropriate cryptographic algorithm [2, 8]. There are many cryptographic systems used to provide security services, which are classified into symmetric and asymmetric types. But, most of the traditional cryptosystems cannot be used for secure IoT environments, because the IoT works with constrained devices that have limited battery, power, as well as memory. Hence, the design of efficient and lightweight cryptographic techniques became a challenge to guarantee secure data transmission in the IoT networks [1, 2]. The lightweight cryptography should fit the low energy, computation and memory capabilities of cyber-physical systems. Also, it should provide an optimized security/cost/performance trade-off [1].

There are many researches about providing security to the IoT systems. Several lightweight cryptography methods were proposed by researchers to be used in secure IoT sensors/devices to protect data transferred and collected. In 2017, Usman et al. [8] proposed a lightweight encryption algorithm named secure IoT (SIT). This algorithm is a symmetric block cipher based on mixture of feistel and a uniform substitution -permutation network. It encrypts 64 bits plaintext at a time, so it requires 64 bits for key. The 64 bits plaintext is divided into 4 segments, each of size 16 bits, and this algorithm uses five rounds, so it requires five keys, each of size 16 bits; therefore, the 64 bits key is processed using permutation and an f-function operations to construct five keys. The SIT operations used in each round is swapping, bitwise XNOR with key, f-function and finally bitwise XOR operation. The hardware implementation of this algorithm is done on a low cost 8 bit microcontroller. In 2018, Chowdhury et al. [9] proposed modifications on the AES algorithm to make it lightweight and applicable with IoT environment. Their system focuses on using 1D S-Box, which is generated using GF (2^4) instead of using the GF (2^8) in generating the standard AES S-Box. From the implementation results, this proposal is more efficient than the original AES by around 18.35%; the energy consumption when using this proposal is less than that for the original AES and therefore it applies to the IoT environment. In 2018, Tsai et al. [5] proposed the low-power consumption scheme with high security, named the Secure Low Power Communication (SeLPC) method. In this method, some modifications are made to AES algorithm; the first modification is decreasing the encryption cycles of AES for reducing end-devices data encryption power. The second modification is using the encryption key and D-Box update procedure (the Dynamic Box (D-Box) instead of S-Box in AES) to enhance security level. Also, the simplification of AES encryption processing can reduce power consumption. The analysis results showed that the SeLPC can avoid many attacks also minimized the encryption power by up to 26.2% compared with the original AES. However, it was used with IoT environments. In 2018, Hassan and Habeeb[10] suggested a method to build a secure web of things system to manage and monitor the patient information. This proposal provides security to the information collected from IoT sensors. The modified TSFS encryption algorithm was used to provide security in this system. It achieves high efficiency with lower cost, and the time is not affected when compared to other systems. The hardware devices used in this proposal are Arduino and Raspberry Pi, with different types of sensors. Each microcontroller (Arduino) is connected to different types of

sensors and, through the Arduino microcontroller, these sensors are connected to the Raspberry Pi device to provide security and privacy to the sensors data. In 2018, Khalaf and Mohammed [11] proposed two types of security service for an IoT smart home application; the first security service is providing the confidentiality which is achieved by the encryption process on all the sensors data sent to IoT server. The encryption was performed using AES-GCM or RSA-OAEP encryption algorithm. The second security service in that work is the integrity, which is achieved by using SHA3-512 algorithm. The hardware devices used to implement the smart home application are Raspberry Pi 3 and four sensors. The evaluation results showed that their system takes a shorter average time of about 3ms when using AES-GCM algorithm, whereas the RSA-OAEP algorithm takes about 9ms. Furthermore, the integrity service takes about 25% of encryption time. In 2019, Naif et al. [12] proposed modifications on the AES algorithm to make it lightweight. This proposal uses the same operations as the original AES, except for the MixColumns operation which is replaced by multi XOR stages, shift-cycle operations, and SHA3-256. In that proposal, a new chaotic system (named JORN) is used to generate chaos keys which are used in the encryption process and the calculation of the number of shift cycles and of AES rounds. MLAES uses two S-Boxes, each of size 64 bits. At each iteration, the S-Boxes are shifted by K1 to generate new S-Box values. In that proposal, 40 sensors were used in implementing the MLAES algorithm. These sensors were grouped into 10 groups, each group consisted of four different types of sensors controlled by Raspberry Pi of type B. From the results, it was concluded that the MLAES decreases time consumption and CPU cycles when compared with the original AES.

In the present paper, the lightweight AES algorithm is used in providing security to IoT networks. Lightweight AES (LAES) is a modification on AES algorithm layers, such as S-Box, Keys, and Shifting values, to be based on different chaotic systems. The IoT hardware components used in this work are Raspberry Pi device and sensors. The aim of this work is to protect IoT sensors data, such as temperature, humidity, and flame fire sensors, by the encryption process using the LAES algorithm before sending these data through the network.

2. IoT Architecture

In order to build any IoT project, four basic components are required; these are sensors, processors, gateways, and applications. Each of these tools must have its own characteristics in order to form a useful IoT system [13]. According to the design of other architectures, there is not a consensus on the number of layers for the IoT architecture. Different researches have proposed architectures for IoT systems that are consisting of three, four, or five layers [14]. Several researchers proved that any IoT architecture is consisting of the three main layers: Perception (sensing) layer, Network layer, and Application Layer [4, 14]. The simple architecture of IoT system may consist of the following four layers: sensing (Things (IoT devices)), network, service, and interface.

2.1. Sensors

The sensors represent the front end devices and the key building blocks of the Internet of Things. These sensors are also called "Things" of the system; their main goal is to collect data from its physical environment. They should have unique identifiable devices with a unique IP address so that they can be easily identifiable over a large network. In addition, they must be active in nature to be able to collect real time data. These devices can work either on their own (autonomous in nature) or made to work by the user. Examples of sensors are gas sensor, water quality sensor, moisture sensor etc. [13, 15]. Each type of these sensors converts signals of physical parameters, temperature, humidity, motion, etc., to digital or analog form that can be readable by the machines and humans [15, 16].

2.2. Processors

In any IoT system, the processors represent the brain of the system. Their main function is to process the collected sensing data to extract the valuable data from the enormous amount of raw data collected. Processors mostly work on real-time basis and can be easily controlled by applications. These are also responsible for securing the sensing data by performing encryption/decryption. Embedded hardware devices, microcontroller etc. are the ones that can process the data because they have processors within [13]. The most famous devices used in IoT systems are Arduino and Raspberry Pi.

Arduino: Arduino is a microcontroller device that is easy to use. It is connected to a computer and can run a single program at a time. Arduino is an open source platform, meaning that the hardware is

reasonably priced and the software development is free [10]. Arduino boards are able to read analog or digital input signals from different sensors and convert them into an output, such as activating a motor, turning LED on/off, connecting to the cloud and many other actions depending on the system. There are various types of Arduino devices, the most famous of which is the Arduino UNO [16, 17, 8].

Raspberry Pi: Raspberry Pi is a mini computer with an operating system. It can run multiple programs at a time. One of the Linux option Debian, called Raspbian operating system, is a great match for Raspberry Pi because it is free and open source, keeping the price of the platform low [10, 19]. Raspberry Pi is considered as the main component of the IoT concept. There are various generations of Raspberry, such as Raspberry Pi Zero, Model A, Model B, Model B+, etc. [16].

2.3. Gateways

The processed data are sent to the proper location using gateways. Therefore, The gateways are responsible for routing the data. In other words, the gateway helps in communication and provides network connectivity to the data. The network connectivity is very important for any IoT system in order to communicate. LAN, WAN, PAN etc. are examples of network gateways [13].

2.4. Applications

Applications form another end of an IoT system. They are essential for proper utilization of all collected data. Applications are controlled by users and represent the delivery point of particular services. Some examples of applications are: home automation apps, security systems, industrial control hub, etc. [13].

3. IoT security

Data security has become an essential requirement for various organizations. Different entities communicate with each other through networks or the Internet. Thus, a secure communication must be provided [18-22]. In recent years, the IoT played an important role in different fields of our life, due to utilizing different connected devices that can sense, compute and exchange data through networks or the Internet. Because of the sensitivity of applications in the IoT network, the security of the transmitted data is very important, There are many encryption algorithms [2], but these algorithms often delay the communications due to the additional operations used [23]. Therefore, to protect IoT sensors data, a lightweight encryption algorithm became a requirement [24]. The design of lightweight cryptosystem has become a great challenge that the designer needs to deal with the trade-off between achieving robust security level with low cost and enhanced performance. Three factors are required for implementing the lightweight cryptography: size, power consumption and speed (throughput/delay). The power consumption is important with those devices operated with limited battery supplement. Also, a high throughput is necessary for devices that transmit large data such as a camera or a sensor, while a low delay is important for the real-time control processing such as a car-control system, etc. [25]. There are many lightweight cryptography systems, such as PRESENT[26], CLEFIA[27], KATAN[28], HEIGHT[27], SIMON/SPECK[29], TEA[2], TSFS[30] and many others. The lightweight cryptography aims to provide enough security levels with optimum use of resources [10]. Some block cipher algorithms are summarized in Table-1.

Table 1- A Comparison between different lightweight block cipher algorithms

Algorithm	Block size (bits)	Key size (bits)	No. of rounds	Algorithm design pattern
HIGHT[27]	64	128	32	GFN
Pickolo[27]	64	80/128	25/31	GFN
PRESENT[26]	64	80/128	31	SPN
DES[2]	64	56	16	Feistel
AES[12]	128	128/192/256	10/12/14	SPN
Clefia[27]	128	128/192/256	18/22/26	GFN
TEA[2]	64	128	64	Feistel
XTEA[2]	64	128	64	Feistel
NTSA[2]	64	128	64	Feistel

There are different approaches to develop lightweight cryptography, such as modifying existing algorithms, optimizing existing algorithms, or developing new algorithms with lightweight issues [27].

4. The Proposed IoT security method

The main goal of this proposal is to use a lightweight, secure, and fast symmetric encryption algorithm (LAES) in protecting IoT sensors data. The LAES algorithm represents a modification on AES algorithm using chaotic systems. Due to the high compatibility between the chaotic systems and cryptography, LAES algorithm depends on different chaotic systems in designing all its needed tools, such as S-Box, keys, tables of permutations and shifting values. The LAES algorithm layers are different from the original AES in the following stages: The Initial Permutation (IP) is used instead of the ShiftRows and the dynamic Shift Rows is used instead of MixColumns. Two types of IP tables are used in initial permutation operation, one for odd rounds and the other for even rounds. Also, the S-Box used in SubBytes operation is generated depending on the 1D chaotic logistic map system represented in equation (1):

$$x_{i+1} = \mu * x_i * (1-x_i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

where x_0 is the initial state, i is the number of iterations, and μ is the system control parameter. The value of x_{i+1} is a number between zero and one for all i while the value of the control parameter μ belongs to the interval (0, 4) [31].

The IP tables used in the permutation step of LAES algorithm are generated from the 2D chaotic logistic map system represented in equations (2) and (3):

$$x_{i+1} = \mu_1 * x_i (1- x_i) + \alpha_1 * y_i^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

$$y_{i+1} = \mu_2 * y_i (1- y_i) + \alpha_2 * (x_i^2 + x_i * y_i) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

The initial parameters, $x, y \in (0,1)$ and the system have chaotic behaviours when the control parameters have the following values [32]:

$$2.75 < \mu_1 < 3.4, \quad 2.7 < \mu_2 < 3.45,$$
$$0.15 < \alpha_1 < 0.21, \quad 0.13 < \alpha_2 < 0.15$$

Finally, the keys used in the add round key operation are generated from the 3D chaotic logistic map system shown in equations (4), (5) and (6).

$$x_{i+1} = \mu * x_i * (1- x_i) + \alpha * y_i^2 * x_i + \beta * z_i^3 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

$$y_{i+1} = \mu * y_i * (1- y_i) + \alpha * z_i^2 * y_i + \beta * x_i^3 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5)$$

$$z_{i+1} = \mu * z_i * (1- z_i) + \alpha * x_i^2 * z_i + \beta * y_i^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (6)$$

The three initial parameters, x, y and z belong to (0,1) and the system has the chaotic behaviour when the control parameters are [32, 33]:

$$3.53 < \mu < 3.81, \quad 0 < \alpha < 0.022, \quad 0 < \beta < 0.015$$

Three types of chaotic keys are generated in the first for odd rounds, the second for even rounds, and the third for initial round key. Algorithm 1 represents the generation of all these tools of LAES encryption algorithm.

Algorithm 1: Generating IoT Security system tools

Input: 1D logistic map initial conditions (x_0, μ)

Output: S-Box (16*16), IPO(4*4),IPE(4*4), keys for odd rounds, keys for even rounds, keys for initial rounds, shifting values array of size (40).

Begin

Step1: Read initial values.

Step2: Generate chaotic sequence using 1D logistic map and save this sequence of numbers in *x* // where *x* is an array of floating numbers.

Step3: Apply some processes on *x* array to construct S-Box of size (16*16).

Step4: Send *x* array generated in step2 to the 2D logistic map system.

Step5: Make some processes on *x* array to be in the range of initial conditions and control parameters of 2D logistic map system.

Step6: Use the initial values generated in step5 to generate two chaotic sequences of numbers *x* and *y* using 2D logistic map system.

Step7: Apply some processes on *x* and *y* to generate two IP tables (IPO and IPE), each of size 4*4.

Step8: Send the *x* and *y* sequences generated in step6 to the 3D logistic map system.
Step9: Make some processes on *x* and *y* sequences to be in the range of initial conditions and control parameters of 3D logistic map.
Step10: Use the initial values generated in step9 to generate three chaotic sequences: *x*, *y* and *z* using 3D logistic map system.
Step11: Apply some processes on *x*, *y*, and *z* sequences to generate three types of keys (odd rounds' keys, even rounds' keys, and initial round key) and generate the array of shifting values.

End

Therefore, each round in *LAES* uses the operations SubBytes, Initial Permutation, Dynamic ShiftRows, and Add Round Key, as illustrated in Figure-1.

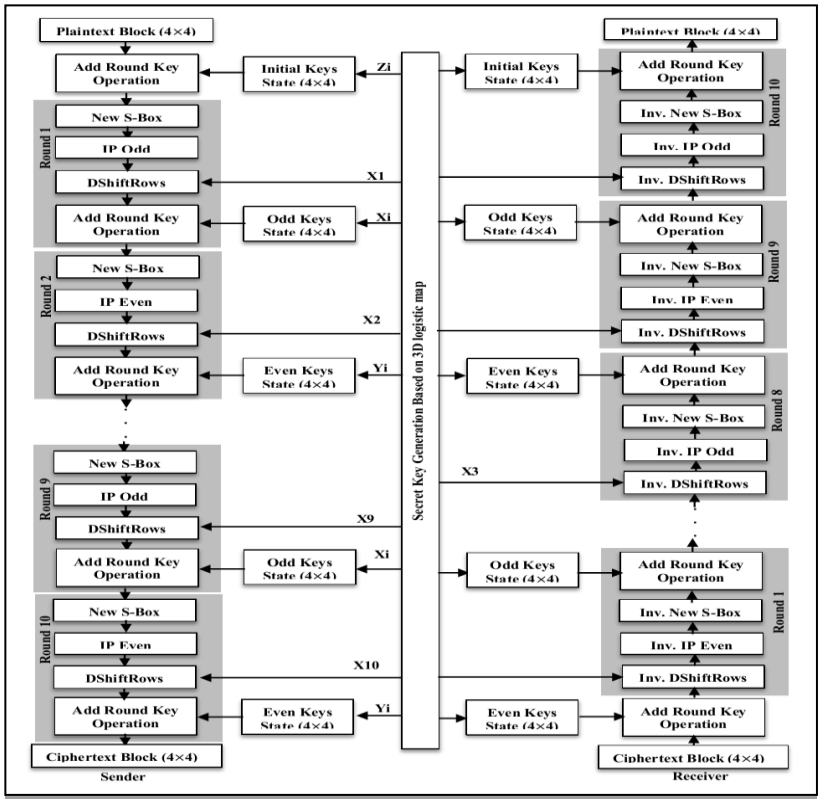

Figure 1- LAES general structure

Odd rounds use an IP table that is different from the IP for even rounds, to increase the diffusion. In the, decryption process the same operations are used but in the reverse order. It uses the operations: Inverse Dynamic ShiftRows, Inverse Initial Permutation, Inverse SubBytes, and Add Round Key.

This proposal is responsible for securing sensors data which are sent through the network. These sensors are used to sense the temperature and humidity, detect if there is a fire or not from using the flame sensor, and send all these data through the network to the server. Hence, these data have important roles in the server's decisions; therefore, the *LAES* encryption algorithm is used to protect these sensors data from unauthorized access and hackers. As a case study, a Raspberry Pi device and different types of sensors are used to monitor temperature and humidity degrees and check if there is a fire or not in the room. In this proposal, the temperature and humidity degrees were measured by the use of DHT11 sensor type, while the flame fire sensor is also used to check if there is fire or not. These sensors data will be sent through the network to the server. These sensors data are very important in the server's decisions. If there is an attacker who knows these data or can change them, this leads to different server's decisions and will make wrong/error in the system. Therefore, these sensors data will be protected through the encryption algorithm (LAES) embedded inside the Raspberry Pi device. This IoT system, for example, can be used in industrial factories, nuclear reactors,

or smart homes to monitor the changes in the temperature and humidity degrees. The first step in the IoT system is to collect the sensing data from sensors that are connected with the Raspberry Pi device. In this proposal, two sensor types were used. These sensors were controlled by Raspberry Pi 3 model B. The sensors deliver numerical data to the Raspberry. In this work, the Raspberry Pi reads data from sensors every five seconds to observe the changes in the environment. After that, the encryption process, using LAES algorithm, will be performed to protect these sensing data. Finally, the encrypted data on the Raspberry Pi will be sent through wireless connection by the socket (IP address) to the server side, as shown in Figure-2. Table- 2 presents the encryption protocol on the Raspberry side.

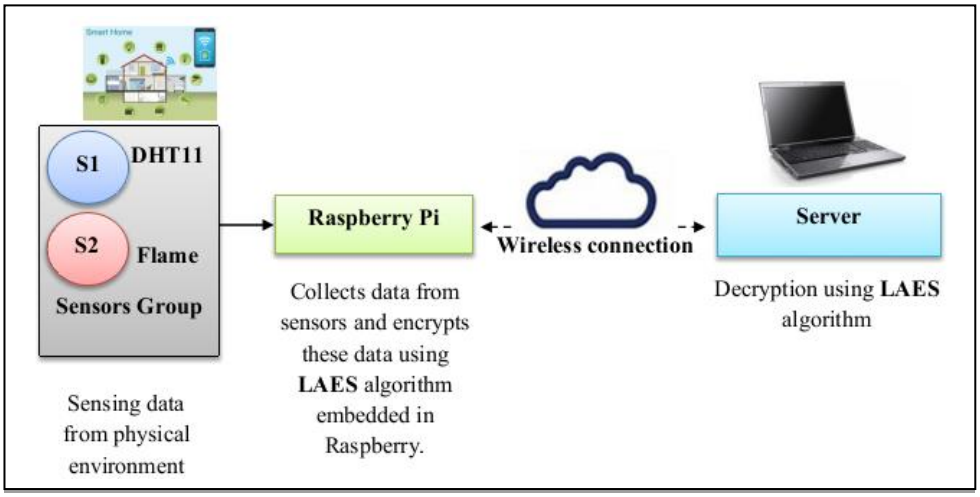

Figure 2- Implementation of the proposed encryption algorithm

In the server side, and after receiving data, the decryption process, using the LAES algorithm, will be performed and the sensors data will be extracted. Table- 3 presents the decryption protocol on the server side. The server used in this proposal is a personal computer that is running windows 10 as an operating system.

Table 2- Encryption Protocol on the Raspberry Side.

Encryption Protocol on the Raspberry Side
1- Start 2- Initialize sensors, raspberry Pi, and network connection. 3- Initialize the chaotic systems. 4- Use chaotic systems to construct LAES tools. 5- Collect sensors data. 6- Apply the LAES algorithm on the data collected from sensors. 7- Send the encrypted data to the server side using socket protocol. 8- If the connection is closed then go to End Else Go to step5 to collect new data from the environment. End

Table 3- Decryption Protocol on the Server Side.

Decryption Protocol on the Server Side
1- Start 2- Initialize the chaotic systems and network connection. 3- Use chaotic systems to construct LAES tools. 4- Read received data from Raspberry Pi. 5- Apply the LAES algorithm on the data packet received, to decrypt these data. 6- Read the received data after the decryption process and print these data on the screen. If these data have unexpected result then makes a simple alert. 7- If the connection is closed then go to End else Go to step4 to receive new data from Raspberry Pi.

End

5. Results and Analysis

The LAES uses the 1D chaotic logistic map to generate an S-Box, that is difficult to guess, to be used in the substitution process. Table- 4 presents the S-Box generated from x_0 =0.201 and μ =3.71. In this proposal, the construction of both the S-Box and its inverse takes only 59.75 milliseconds as average time. The new S-Box is evaluated using S-Box tests criteria, including avalanche, balance, completeness, strict avalanche, and invertibility, as explained bellow.

- ***Balanced Criteria (BC)***

One of the essential S-Box test criteria is to test the distribution of the numbers of 0s and 1s in the output sequences. This distribution should be balanced [34]. Through performing this test, the new S-Box is balanced because it has an equal or near to equal numbers of 0s and 1s. For example, for the word "Computer", after replacement with new data from the S-Box, the output from the S-Box has equal numbers of ones and zeros.

Table 4- S-Box values if x_0 =0.201 and μ =3.71

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	BA	10	97	9B	1F	AC	A1	49	86	BE	D1	FD	5A	AA	0B	4C
1	13	20	B4	95	1D	7A	22	00	D2	F4	F1	85	DC	68	02	1E
2	04	9F	67	3E	A9	12	53	0A	34	36	DF	AF	EF	DB	03	AE
3	EE	3A	2E	E7	E9	76	CA	6B	6F	0D	8D	C7	F6	0E	26	4A
4	AB	4E	61	CF	BC	6A	BD	99	2D	F8	3F	93	14	83	75	5B
5	9C	58	C3	09	3D	92	A5	D9	CD	C6	9A	E4	CE	0C	9D	B0
6	6D	A8	1A	FF	8B	73	50	40	33	E1	D7	A2	42	80	EB	51
7	21	C8	A7	37	D5	79	B1	7B	55	30	84	28	BB	72	69	B6
8	59	FA	39	CB	A0	B8	0F	2C	D3	1B	41	FB	8F	E8	88	23
9	89	EA	8A	05	F7	35	B5	E2	AD	2A	F0	D8	C4	5F	57	06
A	07	D6	47	1C	C5	CC	96	C1	31	01	C9	16	DA	A4	3C	A3
B	98	4B	11	32	DE	D0	2F	56	64	B3	71	F5	25	54	2B	7F
C	45	78	B7	5E	52	FC	C2	63	F2	65	6C	E6	29	91	5C	8E
D	9E	7D	E0	B9	77	60	ED	DD	70	66	18	7C	82	19	FE	27
E	44	87	4D	6E	5D	38	F3	46	81	15	17	3B	62	D4	BF	E5
F	E3	90	8C	43	7E	A6	F9	94	74	C0	08	24	48	EC	4F	B2

- ***The Completeness Criteria (CC)***

This test means that every output bit depends on all input bits. Therefore, if there is one pair of plaintext vectors (z_i and z_{i+1}), where z_i and z_{i+1} are different in only one bit, then the outputs from z_i and z_{i+1} are different at least in a bit k [35]. Each bit of the proposed S-Box depends on the initial values (x_0 and μ) of the 1D logistic map; therefore, if there are two different values of x_0 (they differ in only one digit after the decimal point), then the S-Boxes generated by the first and the second x_0 are different from each other. This property is one of the important features of using chaos. It means that a slight change in initial conditions leads to a massive change in chaotic outputs. For example, the S-Box generated by x_0 = 0.201 and μ =3.71 and the S-Box generated by x_0 = 0.0201 and μ =3.71 are different from each other. Therefore, the S-Box in LAES satisfies the completeness criteria.

- ***Avalanche Criteria (AC)***

The avalanche is the essential criterion in block ciphers, which refers to how a simple change in the input bits leads to a large (avalanche) change in the output sequence; for example, changing one bit from zero to one or vice versa leads to avalanche change in output. This criterion is a desirable feature for block cipher methods because its result is related to the computing of diffusion. The avalanche value should be within the range [0, 1]. The optimal value for avalanche effect is 0.5, which denotes that the avalanche criterion is satisfied [35]. Equation (7) is used to calculate AC.

$$AC = \frac{\text{Number of Flipped Bits in Cipher Text}}{\text{Number of All Bits in Cipher Text}} \quad (7)$$

As an example, when one bit of letter "L" is changed to become "M" and the S-Box (L) is different from the S-Box (M) in five bits from the original eight bits; therefore, the AC is equal to 0.625 which means that it is within the optimal range.

- ***Strict Avalanche Criteria (SAC)***

The S-Box satisfies the strict avalanche criterion if the change of one bit in the input changes half of the output bits. In other words, SAC is achieved if both the completeness and avalanche criteria were satisfied [36]. The S-Box in this paper satisfies these criterions, so the SAC is also achieved.

- ***Invertability***

This test simply ensures that each input value to the S-Box is mapped into a unique output value, forcing the S-Box to be a one-to-one function. This test is necessary to enable the correct recovery (back substitution) which substitutes the values via the Inverse S-Box [36]. The S-box satisfies the invertability feature, if S-Box (L1) = S-Box (L2) such that L1= L2 for all inputs L1 and L2 [35]. Let L1= "a" and L2 = "a", i.e. L1=L2, so the output, in hexadecimal, of S-Box (L1) is "A8" and of S-Box (L2) is "A8". Now, using the S-Box inverse on "A8", the output is " 61" in hexadecimal, which is after that converted to decimal to become "97", which corresponds to char (97) = a. Therefore, the S-Box is invertible due to its ability to extract the original data.

Then, the LAES algorithm is implemented on Raspberry Pi to secure sensors data. The Raspberry Pi collects and aggregates the sensors data, then applies the encryption using LAES algorithm to generate the encrypted data. After that, these encrypted data are sent through the network to the server side. On the server side, the decryption process, using LAES, is performed to extract the original sensors data.

The average time of encryption using LAES algorithm was measured by the timer of Netbeans IDE for java programming. Also, the throughput of the LAES encryption algorithm was measured using equation (8) bellow.

$$\text{Throughput} = \text{Size of Text} / \text{Encryption Time} \quad (8)$$

The LAES algorithm has high throughput and less time in encrypting sensors data on Raspberry Pi, when compared with related work. For example, *LAES* can encrypt text of size of 25 byte in only 1.9285 msec and gives throughput of 12.96 byte/msec, while the original AES takes 2.910 msec and gives throughput of 8.5911 byte/msec. Also, the improvement reported in [10] yields time of 2.30 msec and throughput of 10.869 byte/msec in encrypting the same text size. Table-5 and Figure-3 show this comparison. Also, the total time required for reading data from sensors, encrypting them on raspberry Pi, and sending them to the server was measured. The total time for these operations is only 2.73 seconds.

Table 5- Time of LAES on Raspberry Pi in comparison with related work

Text size (byte)	Original AES (msec)	Throughput (Byte/msec)	Modified LAES[12] (msec)	Throughput (Byte/msec)	The Proposed LAES (msec)	Throughput (Byte/msec)
10	2.908	3.4388	2.19	3.4482	1.021	9.794
25	2.910	8.5911	2.30	10.869	1.9285	12.96
70	3.123	22.414	2.45	28.571	2.307	30.34

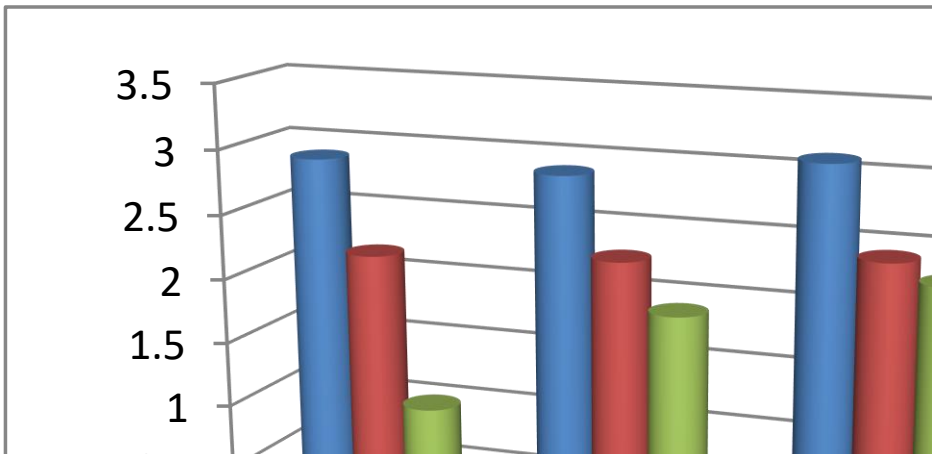

Figure 3- Encryption time comparison with related work

In order to test randomness of the ciphertext, the *NIST* test suite was used, which consists of 16 statistical measurements. They can be used to test the randomness of binary sequences, such as ciphertext or pseudorandom number generators in cryptography. In each of these tests, a *p-value* is computed, which can be used to discover if the test is passed or not. If the *p-value* is greater than or equal to 0.01, then the sequence passes the test. Otherwise, it fails [36]. Table (6) presents the *NIST* statistical test results for the *LAES* algorithm. It shows that *LAES* algorithm passes all the randomness tests and that all these testing results are near to equal one.

Table 6- The results of NIST tests for the proposed LAES

NIST statistical test	Results
1- Block Frequency Test	Pass
2- Frequency(Monobit) Test	Pass
3- FFT Test	Pass
4- Approximate Entropy	Pass
5- Cumulative Sums Test	Pass
6- Serial Test	Pass
7- Runs Test	Pass
8- Longest Runs of One's Test	Pass
9- Overlapping Template of all One's Test	Pass
10- Non- Overlapping Template Test	Pass
11- Linear Complexity Test	Pass
12- Matrix Rank Test	Pass
13- Lempel -ZIV Compression Test	Pass
14- Random Excursions	Pass
15- Random Excursions Variant	Pass
16- Universal Statistical	Pass

The Raspberry Pi CPU status and the memory used during the encryption process are also monitored; Table (7) presents this information. From these results, the LAES algorithm consumes 26.0% of memory size and 1.2% of CPU load of raspberry Pi. It also keeps CPU temperature in a normal state during encrypting sensors data and sending it to the server side.

Table 7- Raspberry Pi status during the encryption process

Raspberry Pi status during running LAES algorithm			**CPU and Memory usage for LAES**
CPU	**Memory**	**Swap**	CPU usage = 1.2% Memory used =240MB (26.0%)
Temp = 48.29 °C CPU average =21.9% Load average =0.27	Total RAM = 926 MB Total used =722 MB Buffers =11.6 MB Cache = 155MB	Total =100.0 MB Used =17.2 MB	

6. Conclusions

In this paper, the *LAES* algorithm was used to secure *IoT* sensors data. The design of the *LAES* algorithm based on multi chaotic systems is a complicated process. Therefore, a slight change in the chaotic initial condition leads to a big change in the output, thus bringing a high level of security. The design of S-Box, IPs, dynamic shift row values, and keys depend on each other, so a small change in the chaos initial conditions leads to a huge change in the design of the S-Box, IPs, values used in dynamic shift rows, and encryption keys in each round of LAES algorithm. The results show that the *LAES* is faster than the original *AES* and the algorithm given in a related work. Besides, it also passes the *NIST* statistical tests. From the implementation on the hardware, the CPU used for this algorithm is only 1.2% and the use of memory is only 26%.

References

1. O. Jallouli, **2017**. " Chaos-based security under real-time and energy constraints for the Internet of Things," Thesis, Signal and Image processing, Universite de Nantes, English, tel-01633910
2. S. Rajesh,V. Paul,V. G. Menon, and M. R. Khosravi **2019**. "A Secure and Efficient Lightweight Symmetric Encryption Scheme for Transfer of Text Files between Embedded IoT Devices," *symmetry mdpi*, **11**(2).
3. M. B. Shemaili, C. Y. Yeun, K. Mubarak, and M. J. Zemerly **2012**" A New Lightweight Hybrid Cryptographic Algorithm for the Internet of Things," *The 7th International Conference for Internet Technology and Secure Transactions (ICITST), IEEE*.
4. J. R. Naif **2019**. "Design and Implementation of Secure IoT for Emergency Response System Using Wireless Sensor Network and Chaotic," *Dissertation*.
5. K.L. Tsai, Y.L. Huang, F.Y. Leu, I. You, Y.L. Huang, and C.H. Tsai **2018**. "AES-128 Based Secure Low Power Communication for LoRaWAN IoT Environments,"*IEEE Access*, Vol. 6.
6. NEC,"Lightweight Cryptography Applicable to Various IoT Devices **2017**." *NEC Technical Journal*, **12**(1).
7. M. Tausif, J. Ferzund, S. Jabbar, and R. Shahzadi **2017**." Towards Designing Efficient Lightweight Ciphers for Internet of Things," *KSII Tranactions on Internet and Information Systems*, **11**(8).
8. M. Usman, I. Ahmed, M.I. Aslam, S. Khan, and U. A. Shah **2017**." SIT: A Lightweight Encryption Algorithm for Secure Internet of Things, "*International Journal of Advanced Computer Science and Applications (IJACSA)*, **8**(1).
9. A. R. Chowdhury, J. Mahmud, A. R. M. Kamal, and M. A. Hamid **2018**. "MAES: Modified Advanced Encryption Standard for Resource Constraint Environments,"*IEEE*.
10. S. Habeeb and R. F. Hassan **2018**. "Build Secure Web of Things system to Manage Patient information Monitoring System," *Iraqi Journal of Information Technology*, **9**(1).
11. R. H. Khalaf and A. H. Mohammed **2018**. "Confidentiality and Integrity of Sensing Data Transmission in IoT Application," *International Journal of Engineering & Technology*, **7**: 240-245.
12. J. R. Naif, G.H. A. Majeed, and A. K. Farhan **2019**." Secure IOT System Based on Chaos-Modified Lightweight AES, " *International Conference on Advanced Science and Engineering (ICOASE)*, 2019.
13. Sukanya **2015**. "A walk through Internet of Things (IoT) basics," *opentechdiary*. https://opentechdiary.wordpress.com/2015/07/16/a-walk-through-internet-of-things-iot-basics-part-2/
14. J. S. F. Hernandez **2018**. "A Comparison of Lightweight Ciphers meeting NIST Lightweight

Cryptography Requirements to the Advanced Encryption Standard," *Thesis*.
15. V. Bhuvaneswari and R. Porkodi **2014**. "The Internet of Things (IoT) Applications and Communication Enabling Technology Standards: An Overview," *International Conference on Intelligent Computing Applications, IEEE*.
16. R. H. Khalaf **2019**. "Secure Mechanisms for Smart Home IoT Application," *Thesis*.
17. Arduino Tutorials point **2016**. " ©*Copyright 2016 by Tutorials Point (I) Pvt. Ltd*.
18. G. Mahalakshmi and M. Vigneshwaran **2017** "IOT Based Home Automation Using Arduino," *International Journal of Engineering and Advanced Research Technology (IJEART)*ISSN: 2454-9290, **3**(8), August,2017.
19. M. Saari, A. Muzaffar bin Baharudin, and S. Hyrynsalmi **2017** "Survey of Prototyping Solutions Utilizing Raspberry Pi," *MIPRO/CTS*.
20. K. Farhan, G.H. A. Majeed, and R.S. Ali **2017**. "Enhancement CAST Block Algorithm to Encrypt Big Data," *Annual Conference on New Trends in Information & Communications Technology Applications, IEEE,* pp. 80-85.
21. K. Farhan and M.A. A. Ali **2017**. "Database Protection System Depend on Modified Hash Function," *In Conference of Cihan University-Erbil on Communication Engineering and Computer Science*.
22. K. Farhan and S. Khalaf **2015**. "New Approach for Security Chatting in Real Time," *International Journal of Emerging Trends &Technology in Computer Science (IJETTCS)*, **4**(3).
A. Ahmad **2018** "A New Security Method for the Internet of Things Based on Ciphering and Deciphering Algorithms," *Kirkuk University Journal/Scientific Studies (KUJSS),* **13**(3).
23. S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park **2017** "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions,"*Springer -Verlag Berlin Heidelberg*.
24. O. Toshihiko **2017**. "Lightweight Cryptography Applicable to Various IoT Devices," *NEC Technical Journal,* **12**(1),Special Issue on IoT That Supports Digital Businesses.
25. L. Dalmasso, F. Bruguier, P. Benoit, and L. Torres, **2019**. "Evaluation of SPN-Basd Lightweight Crypto-Ciphers, "*IEEE Access*, Vol. 7.
26. C.G. Thorat and V.S. Inamdar **2018**. "Implementation of new hybrid lightweight cryptosystem," *Elsevier, Applied Computing and Informatics*.
27. C.D. Canniere, O. Dunkelman, and M. Knezevic **2009**. "KATAN and KTANTAN—A family of small and efficient hardware-oriented block ciphers," *in: Cryptographic Hardware and Embedded Systems, CHES, Springer, LNCS*, pp. 272–288.
28. R. Beaulieu, S. T. Clark, S. Douglas, S. Weeks, B. Smith, and J. Wingers **2013**. "TheSIMON and speck families of lightweight block ciphers," *in: 52nd ACM/EDAC/IEEE Design Automation Conference (DAC),San Francisco*, pp. 1–6.
29. S. Habeeb and R. F. Hassan **2018**. "Sensors data encryption using TSFS Algorithm, " *Journal of Madent Alelem College*, **10**(1).
30. Q. Lu,C. Zhu and G. Wang **2019** ."A Novel S-Box Design Algorithm Based on a New Compound Chaotic System,"*mdpi, entropy*, **21**.
31. K. Farhan and H. Emad **2017**. " Mouse Movement with 3D Chaotic Logistic Maps to Generate Random Numbers, " *Diyala Journal for Pure Sciences*, **13**(3).
32. M. B. Hossain, M. T. Rahman , A B M S. Rahman, and S. Islam **2014** ." a new approach of image encryption using 3D Chaotic map to enhance security of multimedia component", *3rd International Conference on informatics ,electronics and vision ,IEEE*.
33. Maram K and J M Gnanasekar **2016**." Evaluation of Key Dependent S-Box Based Data Security Algorithm using Hamming Distance and Balanced Output," *TEM Journal*, **5**(1).
34. H. Saeed AL-Wattar **2019**." A Review of Block Cipher's S-Boxes Tests Criteria," *Iraqi Journal of Statistical Science First Student Conference*, **29**: 1- 14.
35. N. B. Abdulwahed **2013**." Chaos-Based Advanced Encryption Standard," *Thesis, King Abdullah University of Science and Technology*.
36. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and SanVo **2000**. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Application, " *NIST Special Publication* 800-22.