Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department

Proposed Encryption Algorithm for IoT Video Streaming Security in Building Monitoring

A Thesis

Submitted to the Council of the College of Information Technology at University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology/ Software Department

By:

Nabaa Ali Khalil Ibrahim

Supervised by:

Prof.Dr. Haider Kadhim Hoomod Hashem

2022 AD ***1444 AH***

بسم الله الرحمن الرحيم

وقل رب زدني علما

Supervisor Certification

I certify that the thesis entitled (**Proposed Encryption Algorithm for IoT video Streaming Security in Building Monitoring**) was prepared under my supervision at the department of Software/ College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Master in Information Technology-Software.

Signature:

Supervisor Name: Prof. Dr. Haider Kadhim Hoomod

Date: / / 2022

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "**Proposed Encryption Algorithm for IoT video Streaming Security in Building Monitoring**" for debate by the examination committee.

Signature:

Asst. Prof. Dr. Ahmed Saleem Abbas

Head of Software Department

Date: / / 2022

Certification of the Examination Committee

We hereby certify that we have studied the dissertation entitled (**Proposed Encryption Algorithm for IoT video Streaming Security in Building Monitoring**) presented by the student (**Nabaa Ali Khalil**) and examined him/her in its content and what is related to it, and that, in our opinion, it is adequate with (Viva Result) standing as a thesis for the degree of Master in Information Technology-Software.

Signature:
Name: Firas A. Abdullatif
Title: Professor
Date: / / 2022
(**Chairman**)

Signature:
Name: Dr. Mehdi Ebady Manaa
Title: Assistant Professor
Date: / / 2022
(**Member**)

Signature:
Name: Hiba Mohammed Jaafar
Title: Lecturer
Date: / / 2022
(**Member**)

Signature:
Name: Dr. Haider Khadim Hoomod
Title: Professor
Date: / / 2022
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Dr. Hussein Atiyah Lafta
Title: Professor
Date: / / 2022
(**Dean of Collage of Information Technology**)

Declaration

I hereby declare that this thesis, submitted to University of Babylon in partial fulfillment of requirement for the degree of Master in Information Technology \ Software, has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose sources are appropriately cited in the references.

Dedication

In my opinion, it is my family that deserves thanks for being the main pillar and my source of strength in life. My dear father spent his life in order to reach our ambitions, and his words were like the light that illuminates my path and overcome obstacles. My mother did not miss a corner in prayer without praying for me. My siblings, friends, husband and children I am grateful to all of you. I am grateful for the harsh conditions I went through, which would have ended my undergraduate studies, had it not been for the kindness of God and everyone who supported me at that time.

My thanks to everyone who encouraged me to continue and achieve my dream, even with a word! In the end, I owe it to everyone who believed in my ability to realize my ambitions. I am indebted to my family, my husband and some friends for their constant giving, love and faith in me.

Sincerely,

Nabaa A. Al-Hussainy

Acknowledgment

The real success of a human is the realization of his ambition and the realization of his dream and the possibilities for which God created us. My thanks to God first for success in completing my studies, and I would like to thank the great Messenger of God Muhammad, his brother Imam Ali bin Abi Talib and his pure family, they are like a lamp that lights my way.

And we do not forget to thank the great men who have the greatest credit after credit Allah for the continuation of life in our dear homeland, the martyrs of the Al-hashd Al-shaabi who sacrificed their lives for the sake of Iraq. Their children were orphaned so that our children can live in safety and stability. We will not forget you and your sacrifices will remain an unparalleled honor. Mercy and eternity to their pure souls and The meeting place in when Al-Hussain.

My special thanks, appreciation and respect to my supervisor, Prof. Dr. Haider Kadhim Hmood, for his time, guidance, support and knowledge that he did not spare. My beloved husband who was and still supports me.

Sincerely,

Nabaa A. Al-Hussainy

Abstract

IoT devices are widely used today, and these devices handle digital data like video data streaming. For the importance of this data, a sophisticated security mechanism must be found to prevent breaches of information privacy, since every part of these devices, must be secured to prevent many attacks.

In this thesis, an encryption method for surveillance systems was designed to protect digital data sent over the Internet of Things. The mechanism adopted for the system relies on encryption and authentication to provide security for the data. The system mechanism was designed and built using two lightweight encryption algorithms (hummingbird and speck) and the two algorithms were combined together by hybridization with three different proposed methods. Each proposal is a hybrid lightweight encryption algorithm consisting of parts of the speck algorithm and parts of the hummingbird algorithm, but in a different merging way in each method. The Chaos system was also used to generate random keys for algorithms to make encryption more efficient.

A New proposed modified hashing algorithm was also used to authenticate encrypted data to increase the level of security and to ensure the authenticity and reliability of the data. Because of the weakness in the SHA-3 hash system where it loses the speed required to achieve a fixed-length hash, the design of SFHASH3 has been proposed. This proposal involves incorporating the SPECK - FPHLE algorithm (nine rounds for the purpose of accelerating the encryption process) as an additional layer that complements the layers of the sponge structure found in SHA3.

The proposed lightweight encryption algorithms, it gave satisfactory encryption results, and all three proposals passed all the tests of NIST.

Encryption and decryption time was calculated for each proposed hybrid algorithm on different frame sizes 128x128, 256x256 , 512x512. For the frame size 256x256, the encryption and decryption time was calculated with a different number of rounds each time: 10 rounds, 14 rounds,16 rounds. The encryption results of the first proposed hybrid lightweight algorithm to encryption the average of 7-frame , the 256x256 frame size is 16.024 ms and the decryption time is 15.927 ms. The second proposal gave encryption results for the same previous specifications: 33.81743 ms for encryption and 30.72014 ms for decryption. The results for the third proposal were 13.996 ms for encryption and 13.583 ms for decryption. As shown in the results, the third proposal is the fastest, because the two algorithms work at the same time.

Table of Contents

Table of Abbreviations

Abbreviations	**Descriptions**
AES	Advanced Encryption Standard
CoAP	Constrained Application Protocol
DES	Data Encryption Standard
DoS	Denial of Service
ECC	Elliptic-curve cryptography
HMAC	Hash-based Message Authentication Code
H/W	Hardware
LMAC	Lightweight Message Authentication Code
LWC	Lightweight Cryptography
M2M	Machine to Machine
MAC	Message Authentication Code
MSHA	Modified Secure Hash Algorithm
NIST	National Institute of Standards and Technology
NPCR	Number of changing Pixel Rate
PRNG	Pseudorandom number generator
RAM	Random Access Memory
RFID	Radio-frequency Identification
RGB	Red-Green-Blue

RoI	Region of Interest
RSA	Rivest–Shamir–Adleman
SHA	Secure Hash Algorithm
TEA	Tiny Encryption Algorithm
UACI	Number of changing Pixel Rate
XTEA	eXtended Tiny Encryption Algorithm

Table of Figures

List of Tables

Chapter One

General Introduction

Chapter one
General Introduction

1.1 Introduction

One of the main network models at the moment is the Internet of Things (IoT). The basic idea of the Internet of Things is that there are a large number of devices interconnected with each other, which may be sensors, cameras, mobile devices and many more. Nowadays, the increasing use of the Internet of Things has greatly increased the amount of data and information transmitted in the network. Taking advantage of this technology has become an urgent necessity to keep pace with development and employ it in many fields [1]. Among these important areas are critical building monitoring systems and live video recordings of the venue. Surveillance cameras have developed as in other technologies, and the progress they are witnessing is similar to other modern technical devices, as they are advanced and connected permanently to the Internet to form the so-called Internet of Things technologies [2].

Video streaming is common in IoT networks and has been used in IoT-related monitoring systems. Since it is based on the idea of sharing data between multiple service providers, it is one of the most demanding types of traffic for quality of service and complete security. As some data may be important and confidential and due to the ability of devices connected to the Internet of things to capture it was necessary to maintain its confidentiality [3]. Video content is very important and no one is allowed to access it because it may contain confidential clips. Therefore, it is important to rely on A tight security mechanism for preserving video clips and multimedia that are transmitted through the Internet of Things [4].

Privacy and security are important issues related to the Internet of Things and require more attention and constant research. Recently, the focus was on the important problems associated with the Internet of things in terms of privacy and security due to frequent use of the Internet of Everything in our daily lives [5]. The problem of insecurity in the Internet of Things can lead to users' data being compromised and stolen. Therefore, it was necessary to find encryption algorithms built into these devices. The IoT environment is a very constrained environment with very limited resources. In such an environment, traditional encryption algorithms such as AES cannot be used because they require complex operations. For this reason, a suitable alternative solution has been found which is to use lightweight encryption algorithms to get secure and fast encryption with consideration of cost and performance. Lightweight Cryptography (LWC) is an advanced encryption method used for encryption and increased security at low cost, fewer and simpler computations, and faster processing. Lightweight encryption algorithms are classified into stream ciphers, block and hash ciphers [6][7].

Raspberry pi may be the popular choice when using projects related to the Internet of Things. The camera and the Raspberry pi are used to obtain an encoded video stream . Raspberry pi is a small device that does all the work of a normal computer. It is available at very affordable prices with the possibility to attach its own camera which also transmits the live broadcast and provides many options to the user. The Raspberry Pi can also be connected to a monitor and a mouse and performs many tasks. These tasks are like processing, playing games, playing videos, and much more. More importantly, its memory can be expanded by using external random access memory (RAM) to store information and data. A set of sensors can be connected to a Raspberry pi, which collects information and data from the environment in which it is located and sends this data to the desired destination. Before sending data from these sensors, it goes through an encryption process to ensure that it reaches confidentially and does not violate its privacy from hackers. For this, purpose lightweight encryption algorithms are suitable for working inside different sensors [8].

By making some modifications to the algorithms used, their weaknesses can be eliminated. One possible modification procedure is to use a method to generate keys for the algorithm to make it more robust. Randomization of keys complicates and randomizes the generated keys for cryptographic systems and increases the speed of encryption [9]. A chaotic system is a non-linear dynamic system that exhibits some kind of random behavior that is influenced by the initial conditions. Sensitivity to initial conditions indicates that if a chaotic map is applied to two very closely spaced points repeatedly, this causes the iterations to diverge rapidly and thus become uncorrelated. This is especially useful in image encryption because two adjacent pixels in an image are closely related, but when using a chaotic map they will not be joined after several rounds of iteration [10].

Videos when transmitted over the network are vulnerable to attack. Therefore, using of the chaotic system in encrypted videos helps to increase their protection. This is in that the system avoids leaking the features of people in the video and also helps in resisting a brute force attack. In addition, the generation of a chaotic signal is low cost, which makes it suitable for encrypting big data such as multimedia [11].

1.2 Related Works

In recent years, interest in lightweight encryption algorithms has increased, as they combine fast performance, encryption efficiency, and good security. Many of these algorithms have been introduced.

1- Mohamed et al. (2018) explained the importance of analyzing and processing visual data collected from sensors before they are sent. Because sending it in full size consumes energy and resources, and there is also a lot of unimportant information that doesn't need to be encrypted. Used their own approach to extract key frames from the video to reduce redundancy and increase encryption speed. The frame extraction algorithm used in the work is lightweight, computationally efficient and can be used for small devices such as sensors. Using motion detection can identify important video frames in which motion occurs and is encoded. Relying on using surveillance cameras with multiple visions of the Internet of Things. The advantage of these cameras was the feature of analyzing and examining videos and selecting important data from the unnecessary. So that only a small amount of data is encrypted by a lightweight probabilistic framework encryption algorithm. Encrypting relied on using a 2D chaotic map to create an image-encryption PRNG. In addition to suggesting an RGB image encrypting algorithm to ensure frame privacy as well as using a random approach. The algorithm is based on producing an encrypted image completely different from the original image with the use of the same secret key. Proposing system has passed NPCR and UACI testing and proven resistant to detail attacks on encrypted data. With a frame size of 256 * 256, the encoding speed of this system is 0.1616 (sec) [12].

2- Jawad Kubba & Hoomod.,(2019) proposed to introduce a hybrid algorithm that combines the lightweight cryptographic algorithms PRESENT and Salsa. Used a chaotic system to generate pseudo-random keys. A chaotic system with a two-dimensional map of the chaotic system gives greater complexity to the proposed hybrid algorithm. The suggested algorithm proved to be very efficient and fast in execution time and their random keys passed the NIST random test. The data type proposed for the algorithm is text data collected by sensors. Also, relied on setting a table of the 15 NIST tests and comparing the test results of their proposed algorithm and the PRESENT algorithm. Proposed algorithm took (8.45 Millisecond) to execute while the original PRESENT algorithm took (10.13 Millisecond) to execute on the same data size [13].

3- Naif et al.,(2019) proposed a security system that consists of a modified AES algorithm and combines it with a chaotic 5D system to protect the data of IoT sensors. Also for authentication, hash technologies (SHAKE128bit and .HMAC 256-bit) were used. Since the AES algorithm is complex, they modified it to reduce processing time and increase its speed by about 145%. Chaotic system was used to generate lightweight, modified random AES keys. The system passed the statistical tests passed by the original AES algorithm. The working environment used consisted of 40 sensors connected to a Raspberry Pi B after obtaining the data, it is encrypted and transmitted over the network. statistics indicate that the encryption time of their modified algorithm (132.672 Millisecond) is faster than the original algorithm (161.2235 Millisecond) for the same data size [14].

4- Abdul-Kadhim's (2020) research is based on image encryption using a modified and lightweight current algorithm. Algorithm keys are generated using Chaos System. Proposed a new method for generating algorithm keys instead of the traditional method of generating original algorithm keys that are subject to parsing and breaking. The random key is generated by the unpredictable and unknown logistics map. Experiments show modified algorithm is better than the original algorithm. The entropy of this work is 7.884 as the average of 5 images is close to the ideal value of 8, while the entropy of the original algorithm was 6.779 for five images. The results also showed the processing time of the proposed algorithm is better than the original algorithm. The time of the original algorithm was 97 ms, while the modified algorithm took

60 ms for five images [15].

5- Al-Husainy`s (2020) work was based on proposing a secure video surveillance model to protect content from unauthorized access. To provide an update to the secure and effective key for video encryption, using Chinese Remainder Theory (CRT) to manage the group key. For encryption and decryption, transparent encryption methods are used that protect data when decrypted in computers that display videos. The model can be applied to a variety of scales of smart city monitoring systems. Also, it is made up of Trusted Authority (TA), media cloud, screen, and camera. The encryption system is characterized by a high speed of 91.47 Mbit / s on average [16].

6- Masood et al.,(2021) proposed a lightweight encryption system that uses a multi-stage encrypting algorithm to encode medical images that merge Chaos Theory, Brownian Motion (BM) and Chutech Chen System (CCS). The correlation between pixels of digital medical images has been reduced by using Chaos Maps (Henon) which generates random numbers. According to their results, work can secure fully encrypted medical images suitable for application in real-time cryptography. Also, the model is scalable for encryption videos and audio. It is clear from research results that the result of the time complexity of the model is that it takes 1.53 seconds of time to encrypt a medical image. This is low compared to the time taken by modern technologies at present [17].

7- Denis & Madhubala.,(2021) presented a new hybrid cipher for the encryption of diagnostic data for medical images. The proposed hybrid system is strategically designed by applying Advanced Encryption Standard (AES) as well as Rivest-Shamir-Adleman (RSA) algorithms. In order to ensure the security of diagnostic data to be included with the RGB channels of the medical cover image. Adaptive Genetic Algorithm is one of the new additions used for Pixel Optimization Tuning (AGA-OPAP) that increases data steganography as well as agnostic attributes. The suggested work was evaluated by performing numerical tests. The results proved that their proposed algorithm has the ability to provide secure transmission of medical data. Claimed that the proposed algorithm passed metrics, such as peak signal-to-noise ratio (PSNR), correlation, structural content (SC), structure similarity (SSIM), entropy, histogram, NPCR, UACI and immutability. Mentioned proposed model also can prevent attacks, such as steganalysis or RS attacks [18].

Table (1.1) Summary of related work

Ref. No.	Technique	Year	Data Type	Attacks against /analysis	Metrics	Encryption time in seconds
[12]	Used a 2D chaotic map to create an image-encryption PRNG, also suggesting an RGB image encryption algorithm.	2018	Video Data	Resistant to detail attacks on encrypted data.	NPCR and UACI testing	0.1616 (sec)
[13]	2D chaotic map and Hybrid algorithm PRESENT and Salsa20	2019	Text Data	Insensitive to statistical attacks	NIST	8.45 Millisecond
[14]	5D chaotic map with modified AES algorithm	2019	Data of IoT sensors	Against Brute-force Attack. no differential linear attacks able to broken it.	Their system passed the statistical tests passed by the original AES algorithm(all NIST statistical tests).	132.672 Millisecond
[15]	Modified lightweight present Algorithm based logistics map	2020	Image Data	Attackers cannot analyze information from encrypted images because they are using a chaotic map.		60 Millisecond for 5 images

[16]	Secure authentication protocol , Remainder Theory (CRT), MAC	2021	Image Data	Can resist man-in-the-middle attacks (MITM) and replay attacks is implemented.		The high speed of 91.47 Mbit / s on average.
[17]	Henon Chaotic Map (HCM) Brownian Motion (BM) and Chutech Chen System (CCS).	2021	Medical Image	Differential attacks,	NIST , (HCAV), (APCA), (CA), (HA), (EA), NIST ,(IE), (NPCR), (UACI), (MSE), (PSNR), (TC).	1.53 seconds
[18]	Encryption Standard (AES) and Rivest–Shamir–Adleman (RSA) and Genetic Algorithm	2021	Medical Image	Steg analysis , RS attacks.	PSNR, correlation, structural content (SC), structure similarity (SSIM), entropy, histogram, NPCR, UACI and embedding capacity.	

1.3 Problem Statement

The Internet of Things contains various data, including multimedia data such as videos. Due to the lack of security and privacy in the IoT environment, the fear of attacking and accessing this data has become a concern for many. For this reason, encryption is required to prevent the content from being recognized even after it has been acquired [19]. There are some issues in the IoT environment that should be given a lot of attention by researchers. The main problems of this thesis relate to a set of Privacy problems and security challenges associated with the Internet of Everything as follows:

1. Internet of Things environment (IoT) is constantly growing and very huge, which in the near future may lead to a significant and unexpected increase in the total volume of data. The biggest challenge is how to secure this huge amount of data and be able to take quick action in real-

time. Concerns have been raised about threats to users and the possibility of inability to provide privacy and security and which may lead to the destruction of IoT services.

2. The most important obstacle in transmitting video streams on IoT devices is video encryption. Therefore, choosing fast and powerful encryption algorithms is very important and has an impact on video throughput. But the security challenges become more difficult as IoT devices operate in an environment with limited resources and limited computing power. Well-known encryption algorithms such as DES and AES cannot be used because they require high power and complex calculations. Therefore, the trend was to choose lightweight encryption algorithms, which have the ability to work in such an environment. Also, choosing the above solutions may not provide an ideal security solution because they may cause real-time communication delays due to reasons such as execution time requirements, running environment and executable code length. For example, the time that the encryption algorithm spends on encryption, decryption, and key distribution operations, may lead to an additional burden on IoT devices.

Thesis suggested a security monitoring system for any IoT application. The proposed system is generally divided into three layers, the first layer is responsible for sensing and collecting data. The second layer is responsible for security, this layer consists of two sub-layers (confidentiality and authentication). The third and final layer is the response layer which is responsible for authenticating and decryption data.

1.4 Thesis Aims

In light of the security threats faced by the Internet of Things, researchers have recently proposed several Lightweight Cryptographic (LWC) algorithms that are commensurate with their limited resources. Encryption protects transmitted data from threats. Moreover, a lot of research has revealed many security attacks on a number of these algorithms [20]. The main motive of this thesis is to build a hybrid chaos-based cipher system capable of withstanding the challenges of limited energy and resources in IoT. Where the majority of research relied on simulation to evaluate the efficiency of the algorithm's work in encryption. Suggested implementing the algorithm in a real working environment and on sensors gives more reliable results.

1.5 Thesis objectives

This thesis has three main aims, including:

1. Building a monitoring system for important buildings that broadcasts live videos of the place, equipped with a set of sensors. It also sends alerts on mobile devices in the event of fires or security breaches.
2. Generating random keys using a chaotic system and applying a nine-dimensional logistic chaotic map that gives greater complexity to the proposed hybrid algorithm.
3. Designing a lightweight hybrid encryption algorithm suitable for working within IoT devices with limited resources.

In order for the mentioned suggestions and modifications to be suitable for working in the Internet of Things environment. They must be light in weight, taking into account the limited resources of IoT devices and wired sensors.

1.6 Significant of Thesis

The main contributions of this thesis focus on finding a security solution to the problem of multimedia encryption in a resource-constrained environment such as the Internet of Things. The system is based on choosing the most suitable lightweight encryption algorithms and integrating certain parts of the algorithms to form a strong hybrid algorithm with high encryption specifications. In this case, several problems arise when developing or improving the algorithm. Sometimes the focus is on achieving one goal, as it is difficult to optimize the algorithm to achieve more than one goal. Security may be compromised by hacking when optimizing, as well as when reducing algorithm operations and reducing the number of rounds may expose data to attacks. For this reason, the trade-off is always between security, speed, and response time and they should all be taken into account. In addition to the hybrid algorithm, a powerful and fast key generation system is chosen to avoid any delays that may occur and cause distortion in the encrypted video. To implement these requirements summarized the contributions of this thesis as follows:

1. Proposing a hybrid model consisting of two lightweight encryption algorithms (speck and Hummingbird).
2. Suggesting use the 9D-chaotic system to generate the key for each round.
3. Encryption of the video only when the object path track, not the entire video, this results in faster encryption. Thus, there will be no additional burden on the algorithm and this will increase its speed.
4. Authentication of encrypted data gives stronger results in terms of data security and protects data from tampering and unauthorized access. In this work, it is proposed to use the hash method that will be modified SHA-3 256.

1.7 Thesis Outline

After the first chapter that present a general introduction to the topic, the thesis is organized as follows:

- Chapter Two: **"Theoretical Background "**, provides a comprehensive description of the basic concepts of the Internet of Things system, security and video streaming in IoT. In addition to explaining the encryption methods, the chaotic system, and the criteria used to test the encryption system.

- Chapter Three: **"The Proposed System Design ",** describes the design of the proposed system and the algorithm of the system stages.
- Chapter Four: **"Proposed System Implementation, Results and Discussion "**, describes the evaluation result of the proposed system.
- Chapter Five: **"Conclusions and Suggestions for Future Works"**.

Chapter Two

Theoretical Background

Chapter Two
Theoretical Background

2.1 Introduction

Lightweight encryption technologies are an important area of IoT security. The challenge in the IoT environment is how to achieve secure encryption with low resource consumption [21]. Lightweight encryption has become the appropriate solution proposed to protect data transmitted through IoT devices and sensors. Although there are powerful traditional cipher algorithms, using them in such an environment is resource consuming, energy-consuming and there is no capacity for complex algorithm operations. Lightweight encryption works in proportion to the available resources and for this reason, there are many lightweight algorithms, each with special features and working under certain conditions [22].

In this chapter, some details about the Internet of Things, its structure and its security challenges are presented. Then discussing some details about video streaming and encryption requirements in the IoT environment. The Chaos system is then defined and used to generate the keys, as the video encryption depends on it. After that, the types of encryption methods are briefly explain, the difference between them, and the advantages and weaknesses of each. Then there is an introduction to the alternative encryption to the usual methods, which is the light encryption, its definition and a statement of its features and some of its algorithms.

2.2 Internet of Thing (IoT)

A network of interconnected devices that have the ability to share data between themselves and the rest of the devices on the network without human intervention is called the Internet of Things. Examples of such physical objects include sensors, smartphones, smart watches, and other electronic devices that are connected to each other [23].

These devices have the ability to collect information from the surrounding environment, analyze it quickly and logically, and take the appropriate action necessary. These devices have unique identifiers and are able to transmit information over the network without human intervention [24]. There are many common examples of the Internet of Things, including tracking devices that are implanted inside the bodies of animals to track them, also the pacemaker that is used to monitor the condition of a patient's heart, smart homes, and many others [25].

The difference between the Internet of Things and the regular Internet may be that it not only connects the phone and the computer, but also all the devices together. Therefore, the refrigerator, heating and washing machine are all linked to each other. The Internet of Things makes these devices more intelligent and aware and enables them to send alerts and take actions to facilitate people's lives. For example, a smartwatch can measure your heart rate. This watch can call an ambulance on its own to help if the heartbeat stops. The person wearing this watch has become an integral element of the Internet of Things and this is the exact concept of IoT technology [26].

Artificial intelligence systems can also be integrated with the Internet of Things to facilitate work and learning, make them smarter, and analyze data faster [27].

It also helps reduce working time for companies and factories, improves the performance of machinery, logistics services, and completes work while reducing human dependence, thus reducing material costs. These matters improve the performance of the company and thus increase the spread and strength of its market, its production and its brand [28].

A technology such as the Internet of things is undoubtedly extremely important and useful in our lives, but at the same time there are drawbacks to any technological revolution and therefore the consequences of using the Internet of things must be known. The advantages of IoT lie in the ability to access information at any time, from anywhere and on any device connected to the network. As well as reducing time, money, human and material consumption, accomplishing work more quickly, and improving the quality of task automation [29].

As for its drawbacks, the more devices are connected and the more data is shared between different devices, the higher the potential for data theft. Also, in complex systems that contain huge data, many devices have to be dealt with. Collecting, processing and managing data from all these devices is extremely difficult. Any error or virus that appears in any of the connected devices infects other devices and may disable or damage them. Interconnected devices have different manufacturers and therefore it is difficult to communicate with each other, due to the lack of an international standard for compatibility with IoT [29].

2.3 IoT Architecture

The IoT architecture is made up of users, sensors, a number of physical objects, communication layers, IoT protocols, cloud services, developers, and business layers. The wide spread of Internet of Things objects has led to the lack of an agreed structure for the Internet of Things globally. There are a variety of IoT architectures proposed by researchers and also by consensus there is a conventional and agreed-upon IoT architecture. This architecture consists of three layers: the perception layer (recognition layer), the network layer, and the application layer [30].

1. Perception Layer

It is the bottom layer of the Internet of Things. This layer consists of various physical objects dedicated to collecting information, which sense the surrounding environment such as devices dedicated to sensing heat in a particular place, or a device dedicated to detecting a places known as GPS. Initially, data from the environment is collected by sensors attached to the device. It can be as simple data as temperature and humidity readings, or it can be as complex and important as broadcast videos from homes and important places [30].

2. Network layer

This layer represents the basis of the Internet of Things, due to its ability to transmit information and data from sensors for analysis and subsequent action. This layer is represented by the Internet, which is the bridge between the devices or applications on the one hand and the last layer on the other (IoT architecture's perception layer) [31].

3. Application layer

It is the highest layer in the structure of the Internet of Things, in which services in it are available according to the needs of the user. This layer bridges the large gap between applications and users and this its primary function. After the process of analyzing the collected data, the end user can make use of this information in different ways. The application can also provide services to the user according to the analyzed data by alerting the user either by e-mail or by sending a text message or an alert message . For example sending a text alert when temperatures are too high in the place. Also, a user may want to watch live clips of their home or even video recordings with an application on the phone or through a web browser [32].

It is worth noting that some researchers divide the structure of the Internet of Things into four layers. Whereas the fourth layer is considered as a support layer such as fog computing, cloud computing, and intelligent computing [33]. Other researchers have a different opinion, as they consider that the architecture of the Internet of Things is composed of five layers. In this architecture, the bottom layer is the object layer, and its responsibility is to collect data from heterogeneous devices, analyze the data, and give a unique number for each of them. The five-layer structure of the Internet of Things is illustrated in Figure (2.1) In this structure, the upper layer, which is the application layer, is divided into three sub-layers according to functions [34].

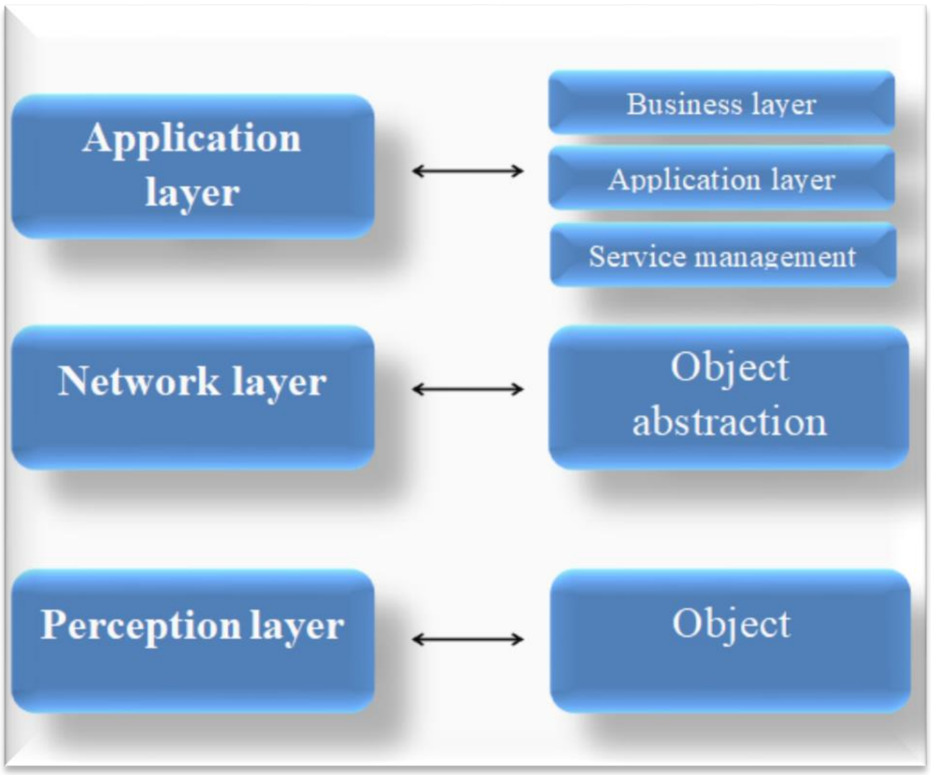


Figure (2.1) The five structural layers of IoT.[34]

2.4 The Security in IoT

The Internet of Things market is expanding day by day as there are millions of devices connected together, every connected device provides opportunities for attackers. The growing popularity of the Internet of Things and the abundance of applications developed for it has coincided with the emergence of security concerns and challenges. The risks are data transmission, accessing devices, broken devices, and always-on/always-off devices. It is normal that there is little interest in issues of security and confidentiality of information transmission by manufacturers. where they are focusing their attention on competing with each other, racing to produce more devices for consumers quickly. With regard to the Internet of Things services, there are many challenges and security issues that need to be addressed and considered [35]. Figure (2.2) presents the most important challenges and security risks that the Internet of Things environment may face.

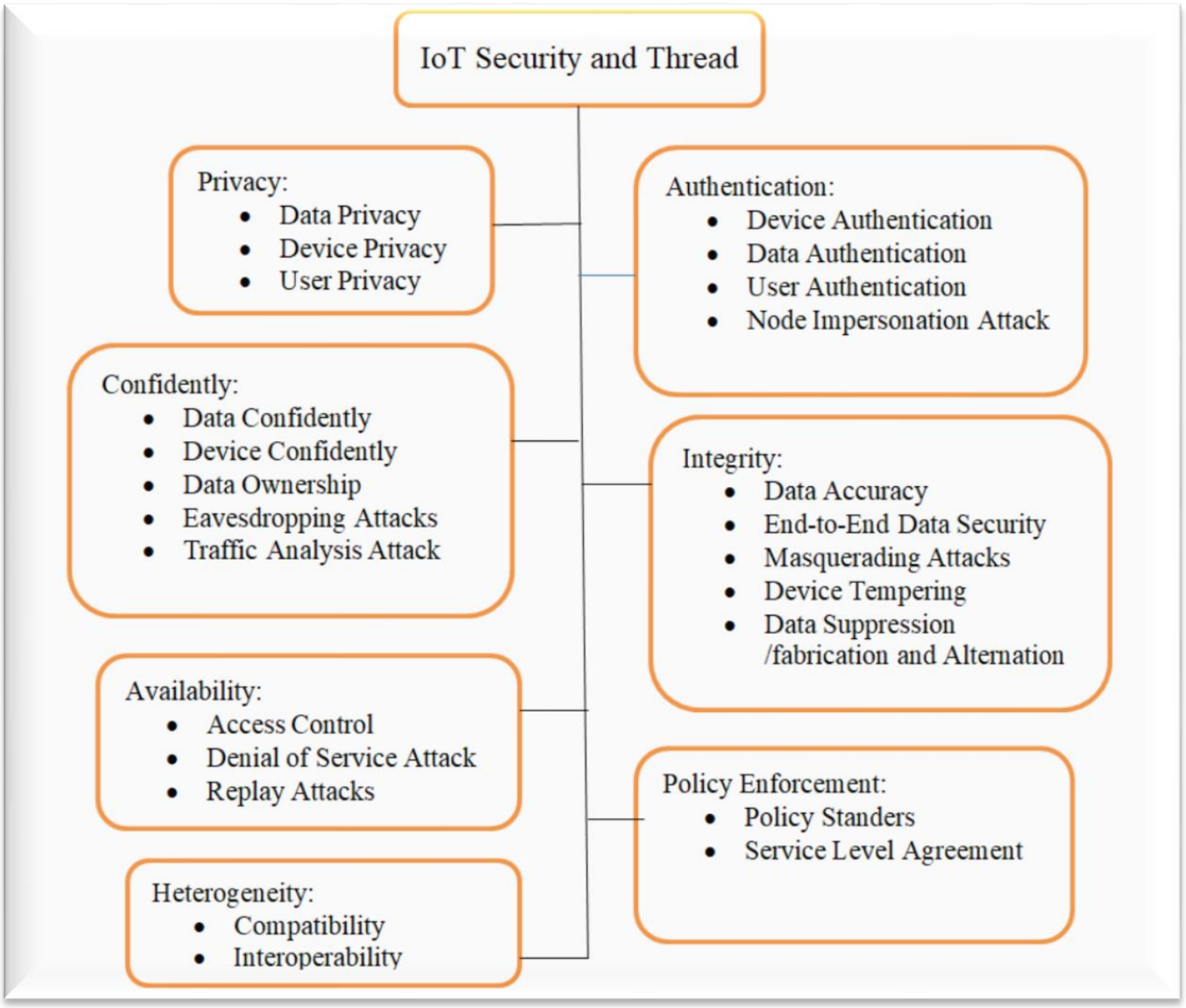


Figure (2.2) Major security challenges and Threats in IoT Environment. [35]

In a multi-layer architecture such as the Internet of Things, a security system for each layer must be built according to the threats facing the layer and the technologies used in it [36]. The Perception layer is very important in the security process of the Internet of Things because it is responsible for collecting information from the environment.

2.4.1 Security challenges in the layers of IoT

There are many challenges for IoT layers as:

1. **Security challenges in perceptron layer**: The characteristics of the perceptron network in the Internet of Things are different from those of traditional network perceptron. This layer contains the sensors that remain in the same place for a long time and are vulnerable to human attack [37]. The responsibility of this layer is to transfer information between IoT devices and this information may be confidential and important which may be attacked by hackers. There are a number of attacks that fall into the perceptron layer that can be described as follows:
   - **Fake nodes attack:** This attack relies on the introduction of malicious or fraudulent nodes between real nodes. These nodes control the connection and can stop the transmission of data and destroy the entire IoT environment. It also allows access, acquisition and manipulation of data [38].
   - **Entering malicious code:** The attacker uses the method of entering malicious code to gain access to the network, take control of it, and cut off services from it [39].
   - **Device jamming attack:** This attack causes much damages, including destroying the contract by replacing the valid parts in the devices with harmful ones. Through this attack, possible to obtain security information related to the encryption and communication keys and the routing table by changing the electronic integration and

obtaining the gate node [39].

- **Attack prevent sleep for nodes:** In the Internet of Things, there are contracts in which batteries are installed when they are located in remote places, so they are programmed to sleep when not in use to save the battery. The attacker prevents these nodes from sleeping and keeping them running by sending wrong instructions to the node, causing damage to the node's battery and thus ending the life of the node [39].
- **IoT service denial:** In this method, the attacker jams the wireless sensor network node by sending noisy signals to the network and jamming the wireless sensor network signals [39].
- **Exploiting data errors:** The probability of an error or deficiency in the transmitted data is very large because the nodes are located at great distances. Wireless transmission of data over these nodes comes with many risks [39].

2. **Security Challenges in the Network Layer:** Transporting data in the network layer requires a challenge in providing security, reliability, and credibility in transit. At the network layer, a number of security challenges appear, namely [40]:

- **DoS attack:** Servers are unable to provide services to users in a denial of service attack. DoS attack causes a complete halt in data transfer between devices [41].
- **Sinkhole attack:** This attack eliminates data security and also corrupted packets instead of delivering them to the correct destination.
- **Man-in-Middle attack:** The attacker can hide his presence in the

network and rely only on the use of a special protocol to communicate with the Internet of Things. The confidential information is obtained through the protocol transmitted between the two sensor nodes [41].

- **RFID authorized access:** Because RFID systems lack security authentication, anyone can access RFID-certified access tags. This means that these tags are easy to manipulate and change [41].
- **Gateway attack:** This attack includes the routing attack, which prevents data from being routed from the Internet to sensors and to nodes. In addition, it works to cut off the connection between the Internet infrastructure and the sensors [41].
- **RFID Spoofing:** The information stamped on the RFID card is obtained by targeting the RFID signal. Thus, through the original identifier, the attacker can transfer his data and gain full access to the IoT system [42].

3. **Security Challenges in the Application Layer:** Hacking the application is very easy if there are deficiencies in security and confidentiality. A virus implanted in the application program code may cause the application to completely destroyed [43]. There are a number of threats to the application layer:

- **Malicious code attacks:** The malicious things that can be injected into the application are used by the code to attack all the applications that have internet service, such as security cameras etc. [44]. For example, a car connected to the Internet of Things was attacked. The attack leads to control of the vehicle's Wi-Fi network, driving control and causing a severe accident.

- **Phishing attacks:** An attacker can impersonate the owner of the email or website by obtaining and infecting them [44].
- **Virus attacks:** Worms, spyware, and viruses can lead to data theft, denial of service, or impersonation [45].

2.5 Concept of the Digital Video

Digital video can be defined as a set of moving visual images (frames) in the form of encoded digital data that can be represented electronically. The digital video consists of a series of orthogonal bitmap images (BMP) that are displayed in quick and steady succession, and appear as a video clip, but are actually collections of images called frames. Each frame consists of a set of pixels, frame rate is defined as the number of frames displayed per second, the more frames displayed per second, the higher the video resolution. The reason for this is that several consecutive and overlapping frames display and capture the finest details of the movement. Man receives information from the world around him through the five senses, and therefore it is analogue information. While digital video and audio information consist of separate units of data. These data are so closely grouped together that it appears to humans that it appears in a constant flow. The frames that make up the digital video are individual, and each of these frames represents a time slice of a scene from the video clip [46].

2.5.1 Video Tracking and object Detection

Video tracking and motion detection technologies are used in surveillance systems to determine the appearance of certain objects. These techniques are divided into methods based on movement, and there are methods based on appearance [47].

1- Motion-based Methods

Background modelling is divided as to the way of data processing into statistical (parametric) approaches and non-statistical (non-parametric) approaches. In the non-statistical methods, initial frame is considered to be the background and the following frames are subtracted from the background. Then the pixels with a value greater than a threshold are considered to be the objects. Each pixel in a frame is considered either a part of the moving object or the background. The background is updated or maintenance along with frame sequences. However, they are suitable for real-time application since it is faster than statistical methods [48].

For the statistical methods, the probability distribution functions (PDF) of the background pixels are estimated. Then in each video frame, the likelihood that a pixel belongs to the background is computed. The statistical approaches have better performance as compared to the non-statistical approaches in background modeling of the outdoor scenes. However, it may require more memory and processing time. In traffic scenarios, for better foreground objects extraction; the impact of dynamic changes of backgrounds should be eliminated well. Such dynamic changes are flag fluttering and swaying trees along the roadside and many other changes [49].

2- Appearance-Based Methods

The objects appearance like vehicles such as (cars, trucks, etc.) differs in size, figure, and color. For detection purposes, segment the foreground and the background methods are employing prior knowledge and using various features such as the color, the symmetry, the horizontal or vertical edges, and the shadow according to the nature of the rectangular shape of targets. Image matching is one of the most important tasks in the target detection and recognition systems. However, the local feature detectors or descriptors methods may be a robust and more reliable where it is designed to allow searching objects' blobs in the consequent images of the frames even when many objects and updating background are considered at the same time. Also, local feature descriptors are invariant to geometric and photometric alteration. Haar-like descriptor [50] for example is a popular descriptor used for car detection in static images and it works based on the edge, line, and the point-surround features and then, many modification approaches for representing the feature and collecting vehicle’s edges. Extra feature detectors work based on other appearance features used to collect contextual information such as FAST corner detector, speed-up robust features SURF , and the scale-invariant feature transform SIFT, BFSIFT, BRIEF, ORB, Local Difference Binary LDB, and KAZE [51].

2.6 IoT video streaming

In recent years, video streaming has been increasingly developed, to provide a better user experience. In order to improve the user experience, the streaming client changes the video quality according to the available conditions of the network and as they change. The video is split into clips from a few seconds of playback, and the video is stored on the server side. The clips are available in multiple bit rates, and adaptive algorithms for the rates are responsible for selecting the bits of the videos. When the client requests information about the stored clip, it chooses the appropriate video transmission quality for the device and its memory size. The video starts playing when the initial buffer threshold is reached. The buffer grows when the available throughput is greater than the encryption rate of the video [52].

In the Internet of Things, there is a need for web monitoring applications for environmental monitoring, such as monitoring a home for healthcare, monitoring of banks or a home for obtaining videos. While the video is uploaded to a server, it is divided into several clips for the client to download. When a certain event occurs, the recorded video clip can be viewed through the camera on the mobile devices of the concerned parties. There may be a feature for the device to select the customer in choosing a specific video display quality depending on the device's capacity or network conditions.

The communication between server and client in IoT is through Constrained Application Protocol (CoAP). It is a protocol proposed as a protocol for transmitting web data in the Internet of Things, as an alternative to HTTP used to transmit multimedia over the Internet. Data can be transferred through CoAP with lower power and bandwidth consumption. CoAP defines a simple mechanism for controlling network congestion and provides reliable communication between IoT end nodes [53].

2.7 Security of Video Streaming in IoT

In the architecture of the Internet of Things layers, multimedia (video and images) is a branch of the perception layer. Multimedia messages and video streams are captured by capture appliances like cameras and transmitted to store servers for processing and then sent over the Internet to the end user. In the process, for the video to reach these repositories, it passes through multiple access points and nodes. Transmission is carried out using heterogeneous communication technologies, so these nodes and access points may be vulnerable to hackers, attacks and espionage [53]. There are many attacks targeting multimedia security in the Internet of Things, there is as follows:

- **Cloning**: The attacker clones and duplicates the multimedia replaces the existing data with a fake and malicious one like a bogus video and thus the user receives fraudulent data [54].
- **Eavesdropping Attack:** While sending data over the network, the eavesdropper is waiting for the data to cross, and as soon as it arrives, he captures the video streams, obtains confidential information, and violates the privacy of the information contained in the video clip [54].
- **Replay Attack:** The attacker records a fake video and then sends the recorded video to the monitoring network. Thus, operators in the control room are deceived that the video is being transmitted in real time [54].
- **Location Disclosure Attack:** Through the video stolen from the network, the attacker reveals information about the place, reference maps and the location of the person. This dangerous information is being exploited by terrorists [54].
- **Unlawful surveillance:** By intercepting surveillance devices connected to the Internet, the attacker can identify the people under surveillance and take information about them [54].

With these threats to IoT video data, it is necessary to have a security mechanism that protects the end-user data from the risks of hacking. On the other hand, the limited capabilities of the Internet of Things make it difficult to encode and transmit videos in real-time quickly and without loss. That is why there was a need to build suitable compression algorithms and techniques for processing and transmitting this information.

Nowadays, a number of security technologies are used for the integrity of multimedia security, such as watermarks, shorthand, encryption and compression [55]. However, these technologies remain insufficient due to the hardware limitations imposed in the IoT environment. There is also a type of encryption called blind encryption, which is widely used to provide security for transmitted data. It encrypts the entire multimedia data (head with payload), which causes a higher computational cost for encryption and an increase in the bandwidth of encrypted data transmission.

Because of these problems, the trend to provide security has been to propose lightweight partial encryption techniques that provide security for multimedia at different levels. In these methods, encryption is applied to the transmitted video, but not all of the video, but only secret parts of it, such as people's faces and license plate numbers. Data obtained from sensors and cameras is partway encrypted and thereafter transmitted through the network. Even if the attacker got hold of the video, it would appear distorted to him.

Lightweight or partial multimedia encryption is convenient and efficient to operate because it has several levels of encryption depending on the computing capabilities of the device. Video encryption may be required when the object is in motion, this will preserve the security of the moving objects in the video. Or it may require background encryption of the video when the background is secret such that the background must not be recognized for location recognition. Encryption of people's facial features is also available to preserve their privacy. Lightweight full video encryption includes maximum security for video and includes object and background motion encryption and face (full partial) encryption [56].

The level of confidentiality is chosen based on the storage capacity, encryption time and transmission of encrypted videos. From conducting experiments by researchers, they found that the use of the AES encryption standard in partial stream encryption provides a maximum degree of security due to its dependence on four rounds of encryption [57].

2.7.1 Requirement of the Video Streaming Encryption

Videos have characteristics such as large data volume, high frequency and the necessity of real-time response. That's why in order for a video encryption algorithm to be perfect, it must meet some requirements :

1. **Real-time requesting:** Encryption and decryption algorithms cannot cause late video access and transmission. For this reason, the encryption and decryption algorithm must be fast to meet the demands of video applications in real-time [58].
2. **Security:** Security is the most important requirement in video encryption. In general, an encryption algorithm can be considered secure if the cost of hacking the algorithm is not less than the cost of licensing paid video content. For example, in broadcasts, content has no value after half an hour of its broadcast. If the attacker can hack the encryption algorithm within half an hour, then this algorithm for this application can be considered secure because it met the security condition at the same time as the broadcast [59].
3. **Withstand transmission errors:** Because real-time video data is transmitted in noisy environments such as wireless channels, the video

that is delivered becomes subject to bit errors. For this reason, the ideal encryption algorithm must be robust to transmission errors and insensitive [60].

4. **Fixed Bitrate and Fixed Compression Ratio:** Fixed compression ratio means that the encryption conversion must preserve the video data size. Constant bit rate is when the constant compression rate algorithm can keep the same storage space or transmission bandwidth [61].
5. **Format compliance:** Often and in many applications, before the video data is sent, it is encrypted or compressed, so the output is data streamlets with some format information. The video data is successfully restored by the decoder which uses the format information for that. It is preferable that the encryption algorithm keeps the multimedia in the same format. This means that multimedia can be decoded with standard decoders without being disabled. This means that when media is encoded with format information, the output will appear random and garbled. When encrypting video data except for the format information, the format of the video information will be preserved. This supports to some extent error strength optimization and also some straightforward operations such as decryption, bitrate conversion, game play, and the like [62].
6. **Low load:** The processing and bandwidth requirements of the encryption techniques should be low because the light load algorithms provide users with a large processing power that enables high-quality video rendering. Furthermore, providing secure streaming should not result in too much bandwidth required for transmission [62].
7. **Security at multiple levels:** The user may sacrifice some amount of

security in order to process a complex video. Encryption systems are available with multiple options of security levels, and person can choose the appropriate level of security for him. It also has the ability to develop in whole or in part by allowing changing the sizes of the keys and repeating the number of rounds. The level of safety is greater with the big size of the key and with the increased number of rounds [62].

2.7.2 Cryptography in Video Streaming

Traditional encryption methods such as AES (encryption), SHA-256 (hashing), and RSA/Elliptic Curve (signature) work efficiently on resource-intensive systems with good processing power and sufficient memory. Although they are efficient and powerful encryption algorithms, they also consume power, storage space, and processing power. Because not enough resources are available for all devices, and because these algorithms cannot be included in them. On this basis, a new type of encryption has been proposed, called lightweight encryption, which targets resource-limited environments such as the RFID environment, sensor networks, and embedded systems such as the Internet of Things (IoT). Lightweight cryptography is a sub-field of symmetric cryptography dedicated to a variety of resource-limited devices in terms of memory capacity, processing power, and limited power resources. There are many lightweight encryption algorithms, such as PRESENT, speck, Hummingbird and many others [23].

Small Scale Embedded Systems are designed using an 8-bit or 16-bit microcontroller. They can be powered by a battery. In this devices it is a difficult to work with traditional encryption algorithms and their requirements in real time. Safety gates available in these environments are few and limited due to the limited power constraints of the device. In lightweight cryptography, the computations are less complex, contain fewer rounds, block size (usually 64-bit or 80-bit), and the key (often less than 90 bits) is smaller than traditional method blocks and keys. There is a major limitation with lightweight encryption. These limitations are related to power requirements, with RFID devices there is no battery connected to the power supply, while the chip operates on power associated with the radio wave [63].

For this reason, an RFID device is limited by the power consumption required for encryption operations, as well as by timing limitations and the number of gates used. Even if the RFID device contains a battery, it can be difficult to recharge it, so it is important to reduce power consumption. While the Internet of Things is widespread, and the availability of security and protection in the IoT is essential, because the success of the Internet everything depends on the percentage of protection available [64].

Lightweight encryption is a compromise that combines security assurance with the use of inexpensive, energy-efficient encryption methods and gateway equivalents. For these reasons, researchers in recent years have relied on developing a lightweight symmetric cipher for the Internet of Things that also includes hash functions, MAC systems such as Quark and Marvin, and block/stream ciphers such as PRESENT, SPONGENT, and others. Asymmetric encryption also has a share in the security of the Internet of Things, including in the theoretical encryption of numbers such as ECC, PBC and the like [65].

In table (2.1) three main characteristics of encryption algorithms and their offerings are presented. As shown in the table, lightweight encryption algorithms must take into account the main characteristics of resource-constrained devices, which are performance, security, and cost. The (physical) properties (performance) can be achieved by applying simple circular functions to small blocks, using small keys, and scheduling simple keys. Security remains the most important among them and is achieved when one of the six internal structures (SPN, FN, GFN, ASX, LFS, Hybrid) is implemented to achieve security against attacks [23].

Table (2.1) Characteristic of LWC [23]

Characteristic		What LWC can offer?
Physical (cost)	Physical Area(GEs, logic blocks)	• Tiny key & block • Simple round with simple computation • Simple key generation
	Memory(registers, RAM, ROM)	
	Battery power(energy, consumption)	
Performance	Computing power(latency, throughput)	
Security	Minimum security strength(bit)	• Strong internal structure
	Attack models(related key, multi-keys)	
	Side-channel and Fault-injection attacks	

2.8 Lightweight Cryptographic Algorithms

There are several types of encryption algorithms, as follows [66]:

1. Lightweight Block Ciphers Algorithms
    a. Lightweight Symmetric Algorithms.
    b. Lightweight Asymmetric Algorithms.
2. Lightweight Stream Ciphers Algorithms.
3. Lightweight Message Authentication codes.
4. Lightweight Hash Functions Algorithms.
5. Lightweight Authenticated Encryption.

2.8.1 Lightweight Block Ciphers Algorithms

Any cipher has the following basic characteristics, the diffusion and confusion that Shannon introduced into cipher systems to increase the strength of cipher. The diffusion in his work is based on the use of substitution (S-box) to complicate the relationship between the cipher text and the key. Whereas the confusion relies on the use of permutation to dissipate the spread of the plaintext statistical structure over the larger cipher text portion. Stream cipher uses only confusion while block cipher uses diffusion and confusion with easier design than stream design. The decryption process in block cipher is difficult while the process is easy in stream cipher to recover the original text [67].

For this reasons, in an Internet of everything is resource-constrained, it is preferable to use block ciphers rather than stream ciphers [63]. Figure (2.3) shows the classification structure of lightweight encryption algorithms. In block cipher, symmetric lightweight cipher blocks use one of the following structures:

- Substitution-Permutation Network (SPN)
- Feistel Network (FN)
- General Feistel Network (GFN)
- Add-Rotate-XOR (ARX)
- NonLinear-Feedback Shift Register (NLFSR)
- Hybrid

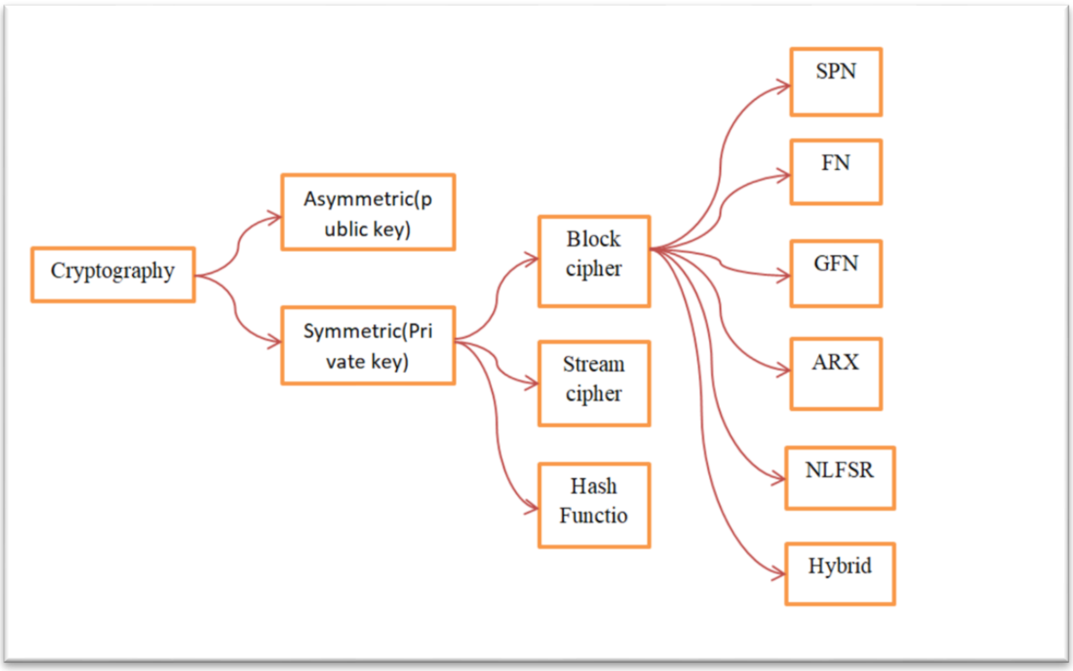


Figure (2.3) classification structure of lightweight encryption algorithms

Substitution-Permutation network: uses a combination of a substitution box and a permutation table to modify the data and reformat it for the next round [23].

Feistel network (FN): The input blocks are divided into two halves of equal size, and the diffusion in each round is applied to only half a round. At the beginning of each round two halves are exchanged.

The generalized Feistel network (GFN): It is a classic version of the Feistel network in which the input block is divided into several sub-blocks, each pair of these sub-blocks to which Feistel functions are applied.

ARX: The S-box is not used in the encryption process. It only uses XOR, Addition, and rotation operations. It is fast and compact, but at the expense of the security it provides compared to SPN and Feistel ciphers.

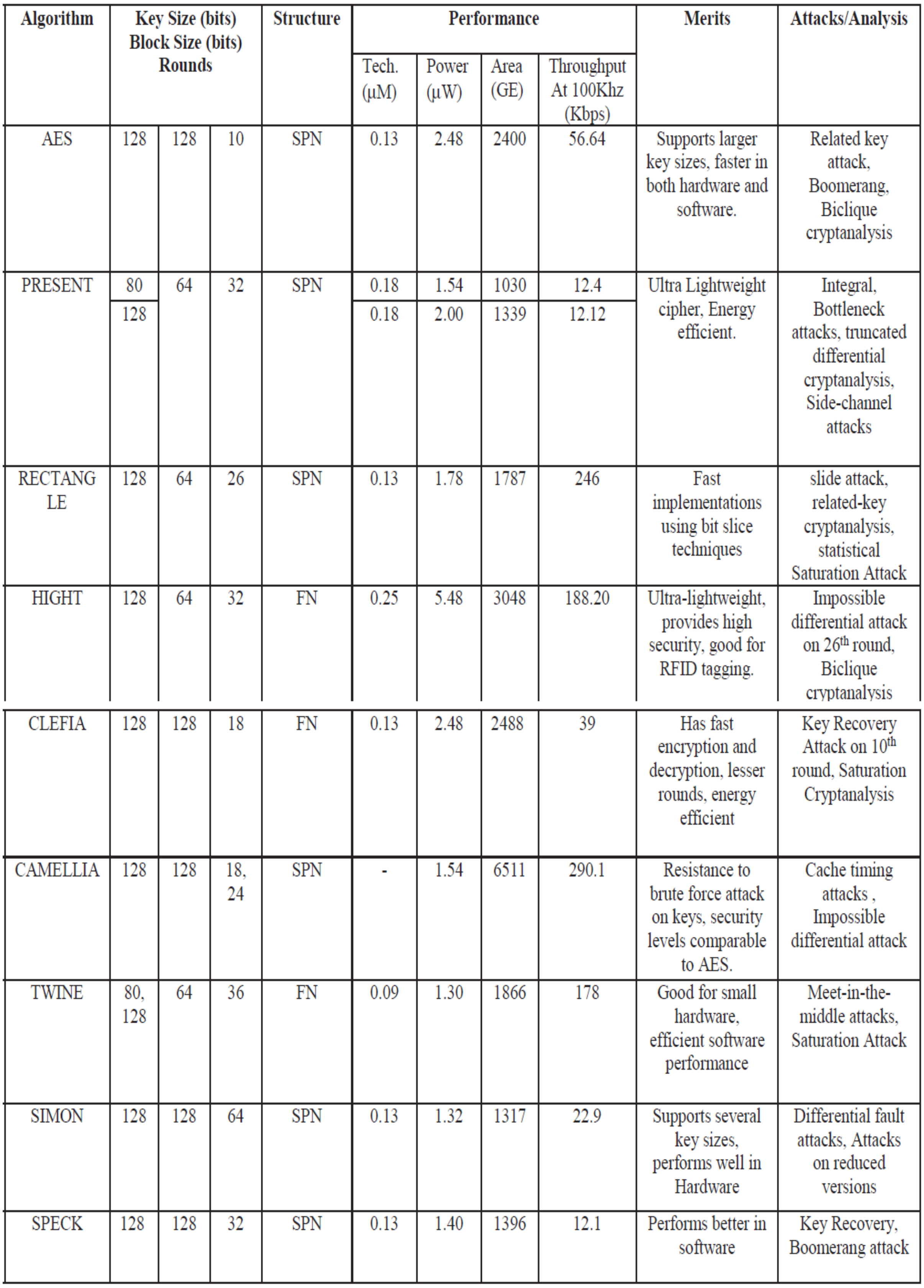
Nonlinear feedback shift register (NLFSR): It works on both types of block cipher and stream cipher. It uses the basis of stream cipher, which is based on taking the current state from the previous state and is a non-linear feedback value.

Hybrid: Hybrid cipher combines any three of the five previous block types. Also, to improve some features (throughput, power, etc.), it combines the features of block and stream ciphers. Table (2.2) shows the classification structure of lightweight encryption algorithms and also shows the class of each algorithm in this classification [23]. Table (2.3) shows a comparison of the performance of block algorithms for lightweight cryptography.

Table (2.2) Structure of lightweight Block cipher algorithms [23]

Structure type	**Algorithm**
SPN	AES, PRESENT, GIFT, SKINNY, Rectangle, Midori, mCrypton, Noekeon, lceberg, Puffin-2, Prince, Pride, Print, Klein, Led, Picaro, Zorro, I-Present, EPCBC
FN	DESL/DESXL, TEA/XTEA/XXTEA, Camellia, Simon, SEA, KASUMI, MIBS, LBlock, ITUbee, FeW, GOST, Robin, Fantomas
GFN	CLEFIA, Piccolo, Twis, Twine, HISEC
ARX	Speck, IDEA, HIGHT, BEST-1, LEA
NLFSR	KeeLog, KATAN, KTANTAN, Halka
Hybrid	Hummingbird, Hummingbird-2, Present-GRP

Table (2.3) Comparison of lightweight block cipher algorithms



Algorithm	Key Size (bits)	Block Size (bits)	Rounds	Structure	Performance				Merits	Attacks/Analysis
					Tech. (µM)	Power (µW)	Area (GE)	Throughput At 100Khz (Kbps)		
AES	128	128	10	SPN	0.13	2.48	2400	56.64	Supports larger key sizes, faster in both hardware and software.	Related key attack, Boomerang, Biclique cryptanalysis
PRESENT	80	64	32	SPN	0.18	1.54	1030	12.4	Ultra Lightweight cipher, Energy efficient.	Integral, Bottleneck attacks, truncated differential cryptanalysis, Side-channel attacks
	128				0.18	2.00	1339	12.12		
RECTANGLE	128	64	26	SPN	0.13	1.78	1787	246	Fast implementations using bit slice techniques	slide attack, related-key cryptanalysis, statistical Saturation Attack
HIGHT	128	64	32	FN	0.25	5.48	3048	188.20	Ultra-lightweight, provides high security, good for RFID tagging.	Impossible differential attack on 26th round, Biclique cryptanalysis
CLEFIA	128	128	18	FN	0.13	2.48	2488	39	Has fast encryption and decryption, lesser rounds, energy efficient	Key Recovery Attack on 10th round, Saturation Cryptanalysis
CAMELLIA	128	128	18, 24	SPN	-	1.54	6511	290.1	Resistance to brute force attack on keys, security levels comparable to AES.	Cache timing attacks , Impossible differential attack
TWINE	80, 128	64	36	FN	0.09	1.30	1866	178	Good for small hardware, efficient software performance	Meet-in-the-middle attacks, Saturation Attack
SIMON	128	128	64	SPN	0.13	1.32	1317	22.9	Supports several key sizes, performs well in Hardware	Differential fault attacks, Attacks on reduced versions
SPECK	128	128	32	SPN	0.13	1.40	1396	12.1	Performs better in software	Key Recovery, Boomerang attack

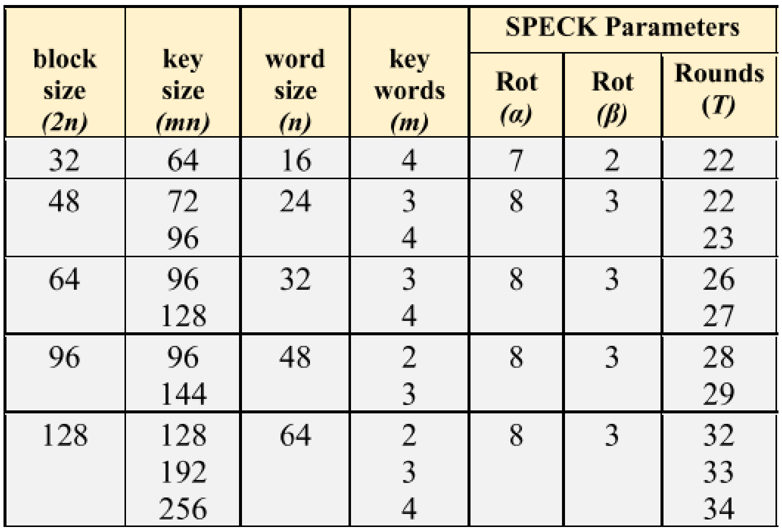
2.8.1.1 Speck Algorithm

It is a family of lightweight symmetrical block blades with a Feistel-like structure. The Speak and Simon algorithm was proposed by the National Security Agency (NSA) in 2013 after traditional cryptography failed to operate within resource-limited environments.

After conducting many experiments, the algorithm has proven to be safe, and it is distinguished from other lightweight ciphers as being more flexible. With the development of devices day by day, no matter what operations they perform, but there is no doubt that they support simple operations such as AND, OR and XOR, which this algorithm operates on, giving the algorithm the advantage to work efficiently and flexibly even in the future [68].

While the rest of the lightweight ciphers work with a fixed-size key and block, Speck's algorithm has the advantage of providing different block and key sizes to suit the user's hardware resources. The algorithm supports 10 different sizes depending on block size (2n) and key size (mn) resulting in 20 different sizes 10 for encryption and 10 for decryption. Table (2.4) shows the variable parameters of the Speck algorithm. To increase the efficiency of the algorithm and prevent rotation attacks and slip attacks, a counter has been placed inside the main Speck scheme [69].

Table. (2.4) Parameters of SPECK [70]



block size $(2n)$	key size (mn)	word size (n)	key words (m)	SPECK Parameters		
				Rot (α)	Rot (β)	Rounds (T)
32	64	16	4	7	2	22
48	72	24	3	8	3	22
	96		4			23
64	96	32	3	8	3	26
	128		4			27
96	96	48	2	8	3	28
	144		3			29
128	128	64	2	8	3	32
	192		3			33
	256		4			34

The round function of Speck uses a number of operations on n-bit words in each round :

- bitwise XOR, $\oplus$,
- addition modulo 2^n, $+$, and
- left and right circular shifts, S^j and S^{-j}, respectively, by j bits.

The SPECK round function for encryption and decryption is denoted by R, as depicted in equations (2.1) and (2.2), respectively. As shown, for both the encryption (equation (2.1)) and decryption (equation (2.2)), the SPECK round function mainly uses XOR ($\bigoplus$) logic operators, addition and subtraction operators, and left-shift (S^j) and right-shift (S^{-j}) round operators, where j is the number of bits being shifted. For both the equations, the rotation amounts of β and α are 2 and 7 respectively, for the block size equal to 32; and for all other block sizes, these two rotation amounts are 3 and 8, respectively.

$$R(x, y) = ((S^{-\alpha} x + y) \oplus k, S^{\beta} y \oplus (S^{-\alpha} x + y) \oplus k) \quad (2.1)$$

$$R^{-1}(x, y) = (S^{\alpha}((x \oplus k) - S^{-\beta}(x \oplus y)), S^{-\beta}(x \oplus y)) \quad (2.2)$$

The block in the Speck is divided into two parts, figure (2.4) and the equations shows the round function of the Speck. containing two parts, the block on the left x (X_i),and the block on the right y (Y_i). k represents one of a set of round keys (k0, k1, . . ., kT−1) where T is the number of rounds. $S^{-\alpha}$ and S^{β} Denotes a left and right circular offset by α or β bits, The products of the i_{th} round are $X_i + 1$ and $Y_i + 1$ [70].

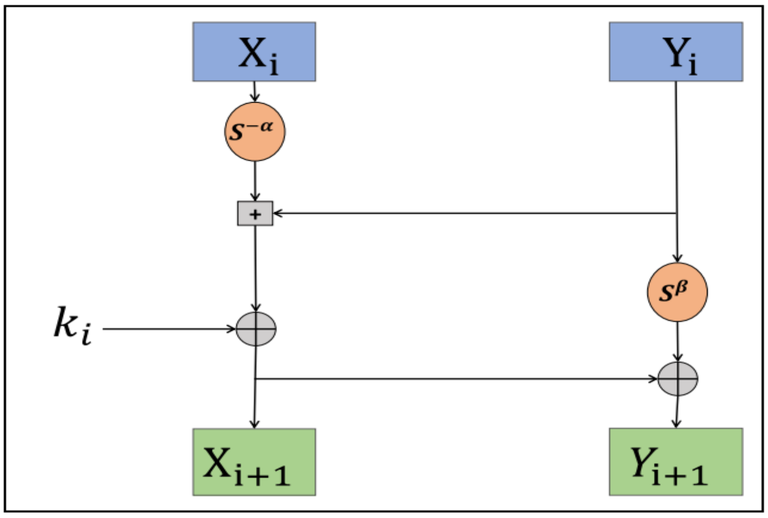


Figure (2.4) Structure of SPECK round function [70]

Speck's key generation functionality requires that round keys be generated from the original k key. The round key is generated for each round using the mentioned key generation function of the round function, the figure (2.5) shows the structure of the Speck key generation function.

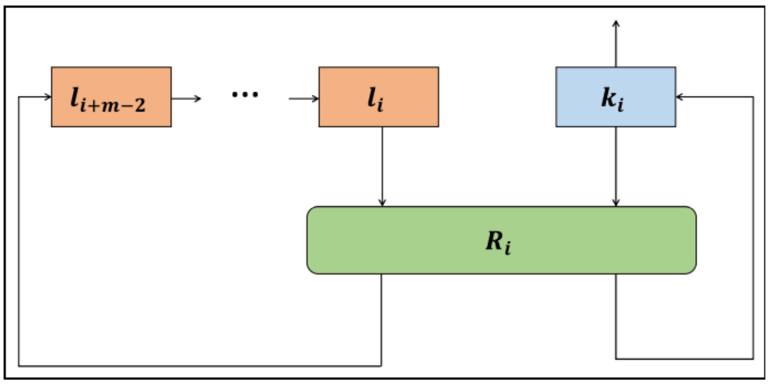


Figure (2.5) Structure of SPECK key generation function [70]

K = (l_{m-2} ,. . . ,l_o ,k_0), m = {2,3,4}. Equation 3 represents the key generation function, where ki is the ith round key, for 0 < i < T:

$$l_{i+m-1} = ((k_i + S^{-\alpha} l_i) \oplus i),\ k_{i+1} = S^{\beta} k_i \oplus l_{i+m-1} \qquad (2.3)$$

Encryption process of Speck: The encryption process in Speck algorithm works in different modes such as ECB, CTR, CBC, PCBC, CFB and OFB. Any data type can be encrypted in the Speck algorithm, including image data. The different cipher modes hide the patterns in the encrypted data by the sequence of outputs from the cipher block or other global deterministic variables in the subsequent cipher block. Table (2.5) shows the different encryption modes that the Speck algorithm works on [71].

Table (2.5) Modes of encryption [71]

Mode		Formulas	Ciphertext
Electronic codebook	(ECB)	Y_i = F(PlainText$_i$, Key)	Y_i
Cipher block chaining	(CBC)	Y_i = PlainText$_i$ XOR Ciphertext$_{i-1}$	F(Y, Key); Ciphertext$_0$ = IV
Propagating CBC	(PCBC)	Y_i = PlainText$_i$ XOR (Ciphertext$_{i-1}$ XOR PlainText$_{i-1}$)	F(Y, Key); Ciphertext$_0$ = IV
Cipher feedback	(CFB)	Y_i = Ciphertext$_{i-1}$	Plaintext XOR F(Y, Key); Ciphertext$_0$ = IV
Output feedback	(OFB)	Y_i = F(Y_{i-1}, Key); Y_0 = F(IV, Key)	Plaintext XOR Y_i
Counter	(CTR)	Y_i = F(IV + $g(i)$, Key); IV = token()	Plaintext XOR Y_i

In the case of image encryption using the Speak algorithm, the image size consists of three parameters M × N × P where M is the number of columns, N is the number of rows and P is the level number (assuming gray scale equals 1). The image is initially stored by " Pixmap ", which stores and displays the image as a rectangular array of pixel colors for easy handling. Algorithm 2.1 shows the pseudocode to implement the SPECK encryption algorithm and Algorithm 2.2 shows the scheduling of SPECK encryption keys [72].

Algorithm 2.1 : Speck Encryption

```
input: plaintext, K encryption key
output: ciphertext
split plaintext into n-bit x, y
    initialize T, β, α
    generate round keys $k_0, . . . , k_{T-1}$
    for i = 0 to T-1
        x = $(S^{-\alpha}x + y) \oplus k_i$
        y = $S^{\beta}y \oplus x$
    end for loop
```

Algorithm 2.2: Round Key generation

```
Input: Encryption key
Output: sequence of keys
initialize m;
generate $l_{m-2}, . . . , l_0, k_0$
for I = 0 to T-2
    $l_{i+m-1} = (k_i + S^{-\alpha} l_i) \oplus i$
    $k_{i+1} = S^{\beta} k_i \oplus l_{i+m-1}$
end for loop
```

2.8.2 Lightweight Stream Ciphers Algorithms

They are symmetric ciphers that generate text encrypted by plaintext data bit streams and associated key streams. The lightweight ciphers are very fast, consume less energy and resources, and provide a high level of security, which makes them the ideal choice for resource-constrained devices and the Internet of Things environment [73].

They are important in applications where plaintext is persistent or of unknown length to secure communication, such as network streams, which are used in military applications [74]. There are many algorithms in this field such as Rivest Cipher 4 (RC4), A5 / 1, E0, Rabbit, HC-128, SOSEMANUK, Salsa20, and many other algorithms. Stream cipher encryption algorithms are briefly pointed in table (2.6).

Table (2.6) The general features of the stream ciphers algorithms [75]

Cipher	Year	Key size(bits)	Block size (bits)	IV	Type	Analysis / Attacks
RC4	1987	8-2048	1	-	ARX	Correlated-key attacks [54,55], attacks on WEP, biased outputs of keystream
A5/1	1989	54, 64	-	0	LFSR	Time-memory tradeoff and known-plaintext attacks, attacks on GSM
E0	1998	128	-	0	SHR	Cryptanalysis , attacks on Bluetooth
AES	1998	128, 192, 256	128	0	SPN	Biclique cryptanalysis

Rabbit	2003	128	128	-	Chaotic tables + simple arithmetic	Practical fault analysis
Trivium	2004	80	1, 8, 16	80	3 SHR	Improved differential fault attack
Salsa	2004	128, 256	32, 512	64, 128	ARX	Attacks on reduced versions
HC	2004	128, 256	-	128, 256	2 Large tables	Cryptanalysis

2.9 Hummingbird Algorithm

It is a lightweight encryption algorithm that is specifically designed to be suitable for devices with limited resources, such as wireless sensors and RFID tags. The algorithm is a combination of block cipher and stream cipher with a 256-bit key size, an 80-bit internal state, and a 16-bit block size. The algorithm has proven its ability to withstand some attacks such as birthday attack, coercive attacks, differential and linear cryptanalysis, cube attacks, and hierarchical attacks [76].

The advantages of the hummingbird are that it is safe and can only provide a small area of the state space for the algorithm. Very suitable for environments with resource constraints such as the Internet of Everything environment. Encryption and decryption also require simple, uncomplicated and few operations, as the algorithm needs a small number of gates. Suitable for implementation on both hardware and software, the algorithm is characterized by its ability to switch keys quickly and easily.

The cipher text cannot be expanded if the message authentication token is not added to the cipher text. The algorithm is based on a rotor machine which is equipped with new rules of rotation and the algorithm structure for initialization and encryption consists of 4 16-bit block ciphers Ek1, Ek2, Ek3, and Ek4. It also consists of 4 internal 16-bit registers RS1, RS2, RS3, RS4 and 16 Linear Shift Feedback Register (LFSR) [77]. For clearness, the details of the encryption used in the algorithm are listed in the table (2.7).

Table (2.7) Notation [77]

PT_i	the i-th 16-bit plaintext block, $i = 1; 2; : : : ; n$
CT_i	the i-th 16-bit ciphertext block, $i = 1; 2; : : : ; n$
K	the 256-bit secret key
$\mathbf{E}_K(\cdot)$	the encryption function of Hummingbird with 256-bit secret key K
$\mathbf{D}_K(\cdot)$	the decryption function of Hummingbird with 256-bit secret key K
k_i	the 64-bit subkey used in the i-th block cipher, $i = 1; 2; 3; 4$, such that $K = k1 \Vert k2 \Vert k3 \Vert k4$
$E_{ki}(\cdot)$	a block cipher encryption algorithm with 16-bit input, 64-bit key ki, and 16-bit output, i.e., $E_{ki} : \{0; 1\}^{16} \times \{0; 1\}^{64} \rightarrow \{0; 1\}^{16}; i = 1; 2; 3; 4$
$D_{ki}(\cdot)$	a block cipher decryption algorithm with 16-bit input, 64-bit key ki, and 16-bit output, i.e., $D_{ki} : \{0; 1\}^{16} \times \{0; 1\}^{64} \rightarrow \{0; 1\}^{16}; i = 1; 2; 3; 4$
RSi	the i-th 16-bit internal state register, $i = 1; 2; 3; 4$
LFSR	a 16-stage Linear Feedback Shift Register with the characteristic polynomial $f(x) = x^{16} + x^{15} + x^{12} + x^{10} + x^7 + x^3 + 1$
$\boxplus$	modulo 2^{16} addition operator
$\boxminus$	modulo 2^{16} subtraction operator
$\oplus$	exclusive-OR (XOR) operator
$m \lll l$	left circular shift operator, which rotates all bits of m to the left by l bits, as if the left and the right ends of m were joined.
$K_i^{(i)}$	the j-th 16-bit key used in the i-th block cipher, $j = 1; 2; 3; 4$, such that $k_i = K_1^{(i)} \Vert K_2^{(i)} \Vert K_3^{(i)} \Vert K_4^{(i)} \Vert$
$S_i(x)$	the i-th 4-bit to 4-bit S-box used in the block cipher, $S_i(x) : F_2^4 \rightarrow F_2^4$
$NONCE_i$	the i-th nonce which is a 16-bit random number, $i = 1, 2, 3, 4$
IV	the 64-bit initial vector, such that $IV = NONCE_1 \Vert NONCE_2 \Vert NONCE_3 \Vert NONCE_4$

A. Initialization Process

When the hummingbird algorithm is implemented in practice, the four RSi internal registers (i = 1, 2 , 3 , 4) are respectively initialized by selecting four 16-bit random NONCEi. This is followed by 4 consecutive operations of encryption on the message RS1 - RS3 by the hummingbird operating in the initialization mode [76].

The figure (2.6, c) shows the initialization process for the hummingbird. To initialize the LFSR a 16-bit final cipher text TV is used. In addition, bit 13 of the LFSR is always set to prevent zero registration. Also, before using the LFSR it is stepped once to update the internal RS3 state record [77]. The algorithm 3 summarizes the initialization process for the hummingbird.

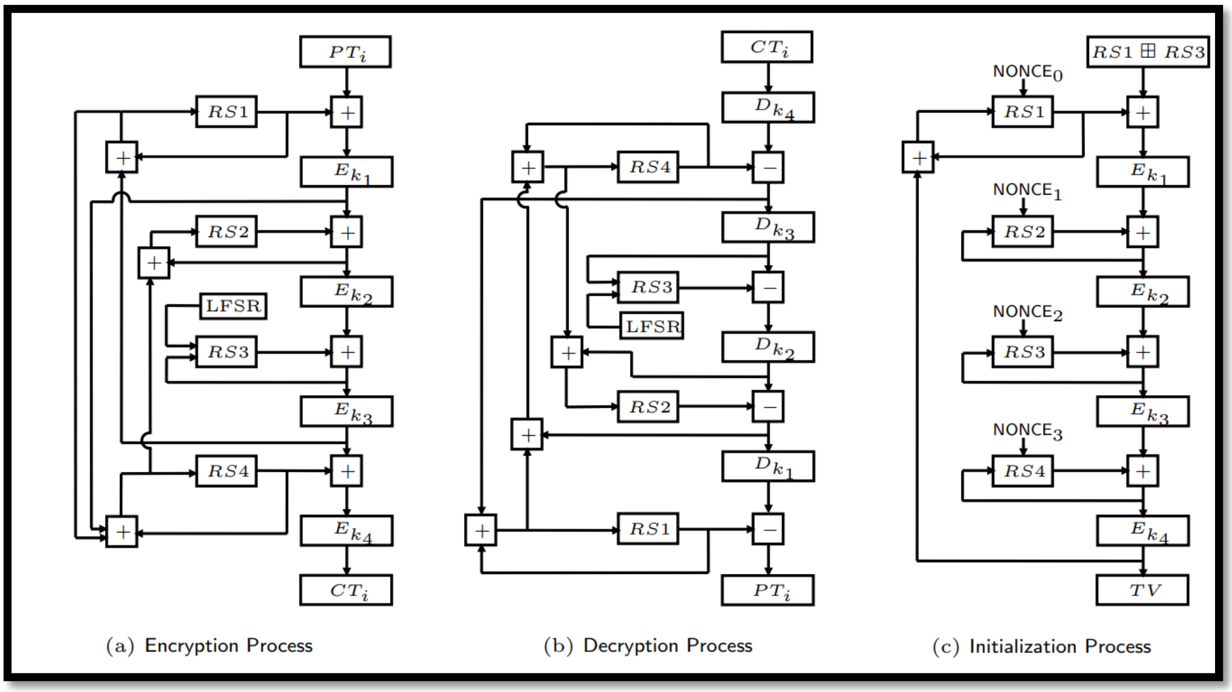


Figure (2.6) The Hummingbird Cryptographic Algorithm and its Internal Structure [77]

Algorithm 2.3 Hummingbird Initialization [78]

```
Input: Four 16-bit random nonce $NONCE_i$ ($i = 1; 2; 3; 4$)
Output: Initialized four rotors $RSi_4$ ($i = 1; 2; 3; 4$) and LFSR
1: $RS1_0 = NONCE_1$                    [Nonce Initialization]
2: $RS2_0 = NONCE_2$
3: $RS3_0 = NONCE_3$
4: $RS4_0 = NONCE_4$
5: for $t = 0$ to 3 do
6:     $V12_t = E_{k1}((RS1_t \boxplus RS3_t) \boxplus RS1_t)$
7:     $V23_t = E_{k2}(V12_t \boxplus RS2_t)$
8:     $V34_t = E_{k3}(V23_t \boxplus RS3_t)$
9:     $TV_t = E_{k4}(V34_t \boxplus RS4_t)$
10:    $RS1_{t+1} = RS1_t \boxplus TV_t$
11:    $RS2_{t+1} = RS2_t \boxplus V12_t$
12:    $RS3_{t+1} = RS3_t \boxplus V23_t$
13:    $RS4_{t+1} = RS4_t \boxplus V34_t$
14: end for
15: $LFSR = TV3 \mid 0x1000$            [LFSR Initialization]
16: return $RSi_4$ ($i = 1; 2; 3; 4$) and LFSR
```

B. Encryption Process

After the system initialization process is completed, the 16-bit PTi plaintext block cipher, by executing a modulo 2^{16} addition of PTi and the content of the first internal state register RS1. Then, via the first block cipher Ek1, the result of the addition is encrypted. Three times the procedure is similarly repeated and the output Ek4 is the corresponding cipher text CTi. Also for the four internal registers and depending on their current state they are updated in an unpredictable way, outputs the first three zeros, and the state of the LFSR [77]. The figure (2.6, a) shows the encryption process for a hummingbird.

Algorithm 2.4 Hummingbird Encryption [78]

```
Input: A 16-bit plaintext $PT_i$ and four rotors $RSi_t$ $(i = 1; 2; 3; 4)$
Output: A 16-bit ciphertext $CT_i$
1: $V\ 12_t = Ek1\ (PTi \boxplus RS1_t)$          [Block Encryption]
2: $V\ 23_t = Ek2\ (V12_t \boxplus RS2_t)$
3: $V\ 34t = Ek3\ (V23_t \boxplus RS3_t)$
4: $CT_i = Ek4\ (V34_t \boxplus RS4t)$
5: $LFSR_{t+1} \leftarrow LFSR_t$                  [Internal State Updating]
6: $RS1_{t+1} = RS1t \boxplus V\ 34t$
7: $RS3_{t+1} = RS3_t \boxplus V\ 23t \boxplus LFSR_{t+1}$
8: $RS4_{t+1} = RS4_t \boxplus V\ 12t \boxplus RS1_{t+1}$
9: $RS2_{t+1} = RS2_t \boxplus V\ 12t \boxplus RS4_{t+1}$
10: return $CT_i$
```

C. Decryption Process

The decryption pattern is similar to the encryption pattern, as shown in the figure [2.6,b]. Also, the process is more detailed in Algorithm 4.

Algorithm 2.5 Decryption Process [78]

```
Input: A 16-bit ciphertext $CT_i$ and four rotors $RSi_t$ $(i = 1; 2; 3; 4)$
Output: A 16-bit plaintext $PT_i$
1: $V34_t = D_{k4}\ (CT_i) \boxminus RS4_t$          [Block Decryption]
2: $V23_t = D_{k3}\ (V34_t) \boxminus RS3_t$
3: $V12_t = D_{k2}\ (V23_t) \boxminus RS2_t$
4: $PT_i = Dk1\ (V12_t) \boxminus RS1_t$
5: $LFSR_{t+1} \leftarrow LFSR_t$                  [Internal State Updating]
6: $RS1_{t+1} = RS1_t \boxminus V34_t$
7: $RS3_{t+1} = RS3_t \boxminus V23t \boxminus LFSR_{t+1}$
8: $RS4_{t+1} = RS4_t \boxminus V12_t \boxminus RS1_{t+1}$
9: $RS2_{t+1} = RS2_t \boxminus V12t \boxminus RS4_{t+1}$
10: return $PT_i$
```

D. 16-Bit Block Cipher

Block cipher is the most important and basic part of hummingbird cipher as it provides protection from a number of attacks such as interpolation attack, differential attack and linear attack. Block cipher encrypts 16-bit data blocks by using a 64-bit sub-key [76].

The block cipher consists of 4 regular rounds and one final round, and each regular round consists of three steps:

- Key exchange step.
- Replacement step.
- The key shuffling step: It involves shuffling the circular key from the data block.

The 64-bit sub-key is divided into 4 16-bit round keys $K_{(1)}^{i}$, $K_{(2)}^{i}$, $K_{(3)}^{i}$, and $K_{(4)}^{i}$ and they are used respectively in the subsequent 4 rounds. From these four round keys two direct keys are derived which are using in the last round $K_{(5)}^{i}$ and $K_{(6)}^{i}$ as shows in figure (2.7). Whereas each of the four regular rounds consists of a master mixing step, a switching layer, and a replacement layer, the final round consists of only the switching step and the S-box replacement steps. A simple exclusive process is used in the key mixing process, whereas the substitution layer is composed of four S-boxes with 4-bit inputs and 4-bit outputs as shown in table (2.8) [79].

Table (2.8) four s box hexadecimal notation [79]

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x)$	8	6	5	F	1	C	A	9	E	B	2	4	7	0	D	3
$S_2(x)$	0	7	E	1	5	B	8	2	3	A	D	6	F	C	4	9
$S_3(x)$	2	E	F	5	C	1	9	A	B	4	6	8	0	7	3	D
$S_4(x)$	0	7	3	4	C	1	A	F	D	E	6	B	2	8	9	5

Linear Transform $L : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$ (2.4)

$L(m) = m \oplus (m \ll 6) \oplus (m \ll 10)$

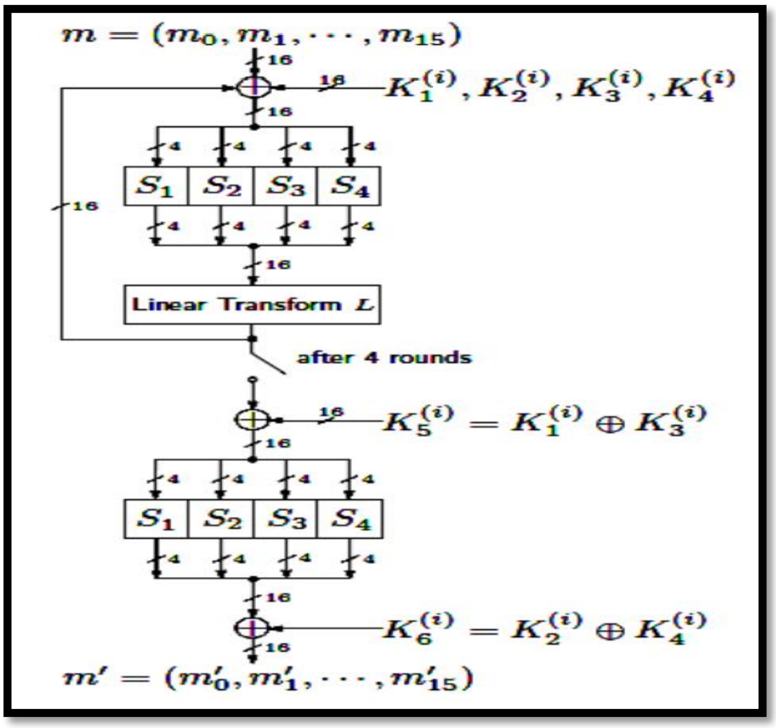

$m = (m_0, m_1, \cdots, m_{15})$
16
16
$K_1^{(i)}, K_2^{(i)}, K_3^{(i)}, K_4^{(i)}$
16
4
S_1
S_2
S_3
S_4
16
Linear Transform L
after 4 rounds
$K_5^{(i)} = K_1^{(i)} \oplus K_3^{(i)}$
$K_6^{(i)} = K_2^{(i)} \oplus K_4^{(i)}$
$m' = (m'_0, m'_1, \cdots, m'_{15})$

Figure (2.7) 16-Bit Block Cipher [79]

2.10 Chaos System

Chaos theory is a modern technique in physical mathematics that deals with highly complex and sensitive systems, where very small changes lead to large results. Chaotic systems are dynamic, complex, and nonlinear systems. Dynamic systems depend on time, and the nature of the system changes as the system continues.

Dynamic systems are categorized into two types, linear and non-linear. The nonlinear dynamic system is characterized by the fact that the output values are not commensurate with the changes that are made to the input. While linear dynamic systems are evaluated through a linear function, it means that the changes that occur in the output are commensurate to the changes that occur in the input. This theory is concerned with studying the properties of deterministic systems whose behavior depends on a set of initial conditions [80].

Chaos theory suggests that, in the long run, moving a butterfly's wings in one place will contribute to changes in the weather by causing very small ripples at first, but over time it will cause a hurricane in another, the example talks about an inevitable causal contribution between the movement of the butterfly and the hurricane, as talking about millions of butterflies around the world may be the cause of this hurricane or even its prevention! The causative event, in turn, may be different, it could be the take-off of a plane or the movement of a baseball bat. The theory assumes that systems that remain in a state of chaos sometimes generate energy, but this energy or its direction cannot be predicted. While the random behavior appears at first, chaotic, stochastic systems can be identified with mathematical equations, which are not without clear order or boundaries. It can be said that the idea of sensitive dependence on the initial conditions and the accumulation of simple effects across the system, the feedback and its permanent link between the inputs and the outputs in continuous renewal [81].

The sequence to the critical points in the systems, the endless branching, and the strange attraction, which combines convergence and divergence, which is the shape of the butterfly in Lorenz's model, nonlinearity and simulation. All of these concepts constitute the dimensions of an integrated model of chaos theory, and despite the scientific complexities of these concepts, they easily cross into the daily life of humans and social systems [80].

2.10.1 Chaos-Based Cryptography

In recent years, there has been a great deal of interest in the study of chaotic systems, which have many advantages, including their dependence on initial conditions, their non-periodic behavior, the similarity of their behavior with that of randomness, and their possession of a continuous broadband frequency spectrum. There are many applications of chaos in many fields, and cryptography was one of them, the so-called chaotic cipher. Where chaos is applied to encryption methods or techniques used in the process of transmitting data in a confidential and secure manner [82].

In order to use chaos theory in cryptography efficiently, entropy is generated by implementing chaotic maps. Entropy produces the propagation and noise necessary for the success of any safety system. The main advantage of using it in encryption is that the chaotic signal looks like noise to users who are not authorized to access its generation mechanism. Also, the sensitivity of the parameters and the sensitivity of the initial state make the initial value and parameter value suitable for generating encryption keys. The performance of any chaotic system in terms of speed and cipher strength depends a lot on the chaotic generator used. Also, the sequences that are generated must show good encryption properties [83].

A generator can be defined as a function that has an input, which is a short random seed, and an output, which is a long stream that is indiscernible from true random bits. The output of these numerical sequences are keys that are used to store and transmit secure data and are necessary to secure communications. This method is fast and inexpensive and depends only on the speed of the processor. However, the sequence of numbers produced in this way is only pseudorandom, so if two similar programs start in the same condition, they beget the identical sequence. Pseudo-random number algorithms based on commas produce non-recursive sequences as if they were truly random. Its advantage lies in the difficulty of predicting the pseudorandom number resulting from the chaotic system due to the lack of statistical results that enable the inference of the random number that results from the chaotic behavior. Chaos TRNs have better random properties than pseudo-random numbers, and they also have unpredictability in the key and cipher [84].

2.10.2 Chaos in the Lorenz system

There are many examples of chaos system , Lorenz`s system which is a simple system and the best-known instance of a chaos system. The Lorenz system consists of a series of three first-order ordinary differential equations. The Lorenz system has become a model that shows how to create very complex behavior from a simple dynamical system [85]. The Lorenz system is defined by the following equations:

$$\frac{dx}{dt} = -ax + ay, \qquad (1)$$

$$\frac{dy}{dt} = rx - y - xz, \qquad (2)$$

$$\frac{dz}{dt} = -bz + xy. \qquad (3)$$

where a, r and b are fixed parameters.

Because the system is nonlinear, there is no analytic explanation for it. At first, the form of the system is converted into an iterative form, and then the numerical solution is calculated. The high-dimensional Lorenz system is used in the encryption to confusion the position of the pixels. The high-dimensional Lorenz system is chosen because its chaotic state is unquestionable [86].

The encryption sequences generated by this system have three main advantages:

1. It is difficult to predict random sequences because the complexity of the system is higher than that of other low-dimensional systems.
2. The possibility of using sequences real value sequence for the three values of system variables individually or in combination.
3. The possibility of expanding the secret key space of the algorithm by setting the three parameters to control the system and the initial state as a secret key. Thus, The chaotic, high-dimensional system has a secret key that is larger than the secret key of low-dimensional systems.

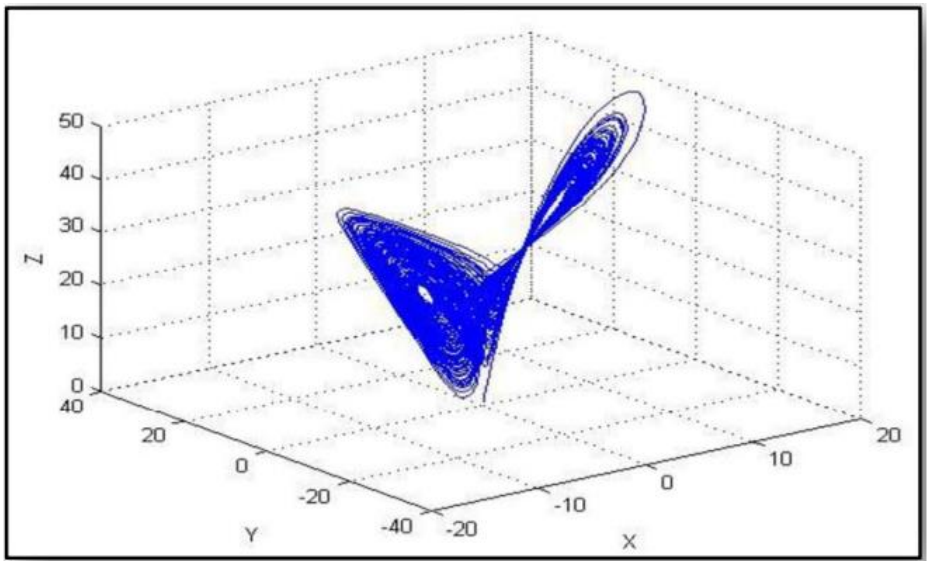


Figure (2.8): Lorenz Chaotic [87]

2.10.3 Chaotic maps

Chaotic maps are used to encrypt video and image data (multimedia) as well as to create pseudorandom number generators. Chaotic maps are the specific formula in which a chaotic system is presented mathematically. Chaotic maps are divided into two types: continuous maps such as the Lorenz system and the Rossler system, and discrete maps such as the logistic map and the Arnold cat map. These maps may be complicated or they may operate with real numbers and have several dimensions, they may be three dimensions, four dimensions, and even six dimensions. The inputs of these chaotic maps are the control parameters and the initial conditions, and their output is a chaotic function [88].

Figure (2.9) shows a scheme for using a chaotic map to generate a series of random numbers. The chaotic series is created using several different random maps. When a map becomes cyclical or deterministic, the chaos sequence must be modified. The final 1-bit pseudorandom output is generated using the binary sequence converter and based on decisions like max, min, minus, compare, and the like [89].

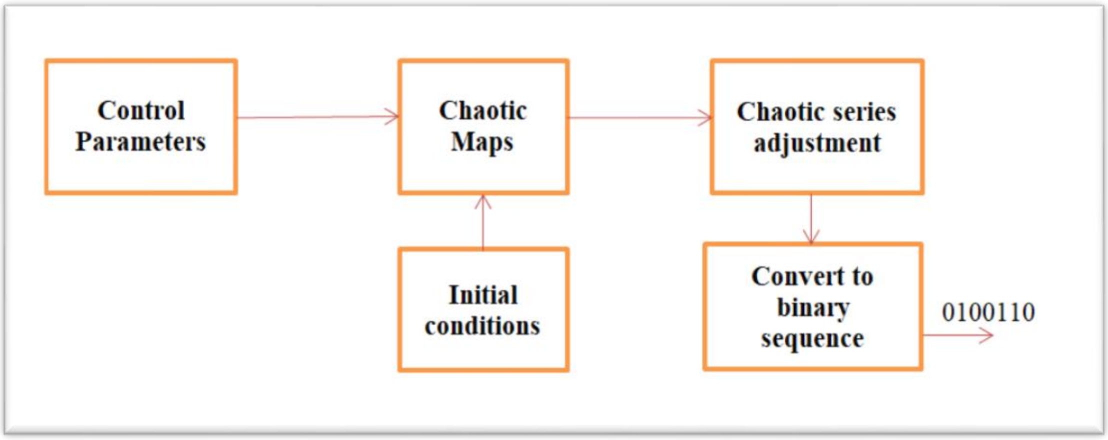


Figure (2.9) General scheme for generating a pseudorandom number using a chaotic map [89].

2.10.4 Video Encryption Using Chaos

The role of chaos-based multimedia encryption has become important and dominant compared to traditional algorithms. Mostly, traditional ciphers such as AES and RSA are used for text encryption, but they are not suitable for video encryption for several reasons, including: video, as know, consists of a large number of frames, consecutive frames are similar, change in a few unit pixels from frame to framework. This massive redundancy of data leads to the failure of traditional methods of masking all visible information.

Moreover, in video encrypting, lightweight encryption is preferred to preserve perceptual information. Because it is possible that traditional ciphers transform the form of data into perceptually unrecognizable content. And because a video consists of a large number of images, each image (frame) in the video is encoded separately. When encrypting a 3D image, it is inserted into the channel separator, the image is converted by the separator into a string of one-dimensional numbers. This is done by transforming RGB channels into separate channels of color, then dividing and adding each row in a single sequence.

The chaotic map type and initial conditions are determined by providing the secret key. When the chaotic map is completed and based on it, the image is encrypted in a confusing sequence and the final encoded image is produced by repeating the diffusion and confusing of a number of N rounds. Sometimes the secret key may affect the number of rounds [90]. Figure (2.10) presents a general scheme for encrypting images based on chaotic maps.

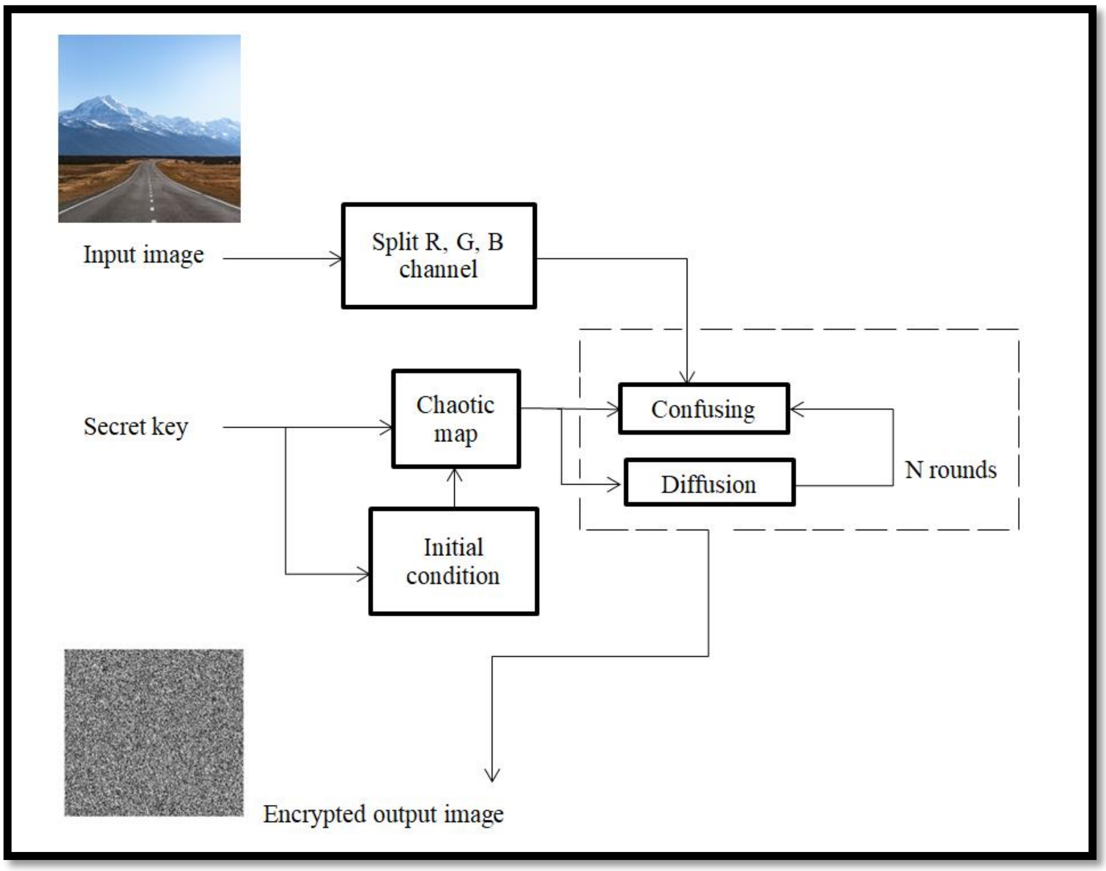


Figure (2.10) General Scheme for encryption images based on chaotic maps [91].

2.11 Message Authentication codes

Authenticated encryption is a form of encryption that achieves encrypted data authentication, integrity and confidentiality simultaneously. message authentication code (MAC) it is a security measure to verify that the received message is indeed from the original source and has not been tampered with or altered. This procedure involves typing a security code by a computer user in order to gain access to the accounts. This code is sent with the user's message requesting access and must be recognized by the MAC receiving system [92].

Authentication can also be achieved through sequence and timing. Usually, three algorithms are used to create Message Authentication Code systems: the key generation algorithm that generates a random key, and the signature algorithm for signing a tag when the message and key are delivered. As well as the verification algorithm to verify the integrity and authenticity of the message when the key and the tag are handed over, and in the event that the message has not been tampered with, it will return an acceptance message and in the opposite case it will be rejected [93].

2.11.1 Hash Function

It is a mathematical function or algorithm that converts big data into smaller data by specifying a small number as the index of the big data in the hash table. The input is random data and the output is a fixed-size encrypted text called the hash value. This ciphertext can then be stored in place of the original text and used to verify the identity of the user. The difference from data compression is that the latter can decompress and restore data, but in the hash function, it cannot return the data to its original size again because it is a one-way function. The hash function can be used to encrypt and decrypt digital signatures. After the process of converting the digital signature by the hash function, the signature and the hash value is sent to the receiving device. The receiving device creates a hash value using the same received hash function and then compares the two functions, if they are the same then the message is from the correct destination and without errors [93]. There are two types of hash functions :

- keyless hash functions (Hash Message Authentication Code HMAC): This type produces a hash value by taking a single parameter (message contents).
- keyed hash functions: This type produces a hash value by taking two

parameters (contents of message and key).

The contents of the message are divided into blocks of equal sizes, and in case the last block is not equal to the other blocks, a padding is added to it. The figure (2.11) shows the hash value creation process [94].

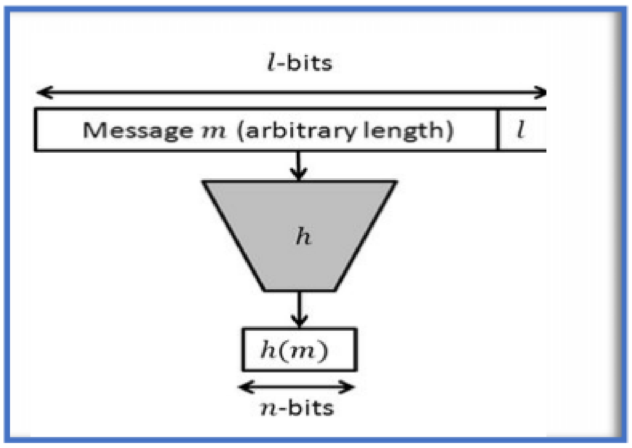


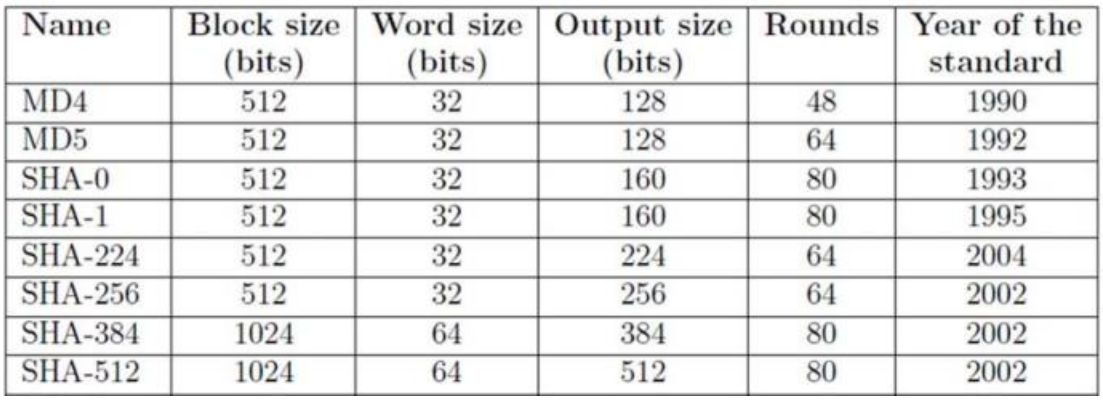
Figure (2.11) process of generating the hash value [95]

The hash functions have several properties, for example, it is not possible to create the original password from the hash or the output. Also, the Diffusion, when changing a small part of the password leads to the production of a very different cipher text. changing half of its hash bits. The deterministic property creates the same hash or encrypted syntax from the same password. Collision prevention determines the difficulty of obtaining two different hashes from the same cipher text (two different passwords). Finally, the unpredictable feature states that the hash value cannot be predicted from the password. One of the advantages of hash functions is that they can slow down the penetration process. Cryptographic hashes take clear text passwords and turn them into enciphered text for storage. Attackers who access the database are forced to decipher those hash values if they want to exploit them. In other words, hashes slow down attackers. Simple cryptographic hashes can slow down attackers, but ultimately attackers will be able to overcome and crack it [96].

2.11.1.1 Secure Hash Algorithm (SHA)

There are many hash algorithms, the most important being the Secure Hash Algorithm (SHA) used for authentication of transmitted messages between the sender and the recipient. After finding significant weaknesses in cryptanalysis, the National Institute of Standards and Technology developed SHA in 1993 and published it as a Standard for Federal Information Procedures (FIPS 180). After finding weaknesses in SHA, known as (SHA-0), the SHA hash function was developed in 1995 with a NIST version called SHA-1 known as SHA-160 and then produced in 2002. It was called NIST FIPS 180-2 It is an updated and modified version of the old version, then new versions appeared, which are SHA-512, SHA-384, and SHA-256, also known as SHA-2. This company added another version in 2002, which is SHA-224. The table (2.9) shows an overview of hash functions, as all older hash versions have the same structure, so they use the same logical and mathematical operations [95,96].

Table (2.9) Standard Hash Functions [95]



Name	Block size (bits)	Word size (bits)	Output size (bits)	Rounds	Year of the standard
MD4	512	32	128	48	1990
MD5	512	32	128	64	1992
SHA-0	512	32	160	80	1993
SHA-1	512	32	160	80	1995
SHA-224	512	32	224	64	2004
SHA-256	512	32	256	64	2002
SHA-384	1024	64	384	80	2002
SHA-512	1024	64	512	80	2002

A. SHA -3

It is the third version of the SHA algorithm and it is also called Keccak. The Keccak framework differs from SHA-2 in that its architecture is devices-oriented and has also relied on hashing growth using a new technology called "sponge". The sponge function gives flexibility to the algorithm structure by allowing the use of inputs and outputs of variable lengths. The domain extender also helps the sponge used to examine the security effects of increasing the message block size to increase efficiency. Keccak was designed for the need for a new cryptographic hash architecture that is more immune to attacks [97].

The message entered in the sponge function is divided into blocks of fixed size, the function depends on four parameters:

1) f = the internal function used to process each input block.
2) r = the size in bits of the input blocks, called the bitrate.
3) C= is referred to as the capacity (complexity of the sponge construction).
4) Pad = the padding algorithm.

The default values for Keccak are c = 1024 bits, r = 576 bits, and b = 1600 bits.

In the figure (2.12) the SHA-3 block diagram is shown, Part B of the diagram shows more details of SHA-3's main algorithms and functions.

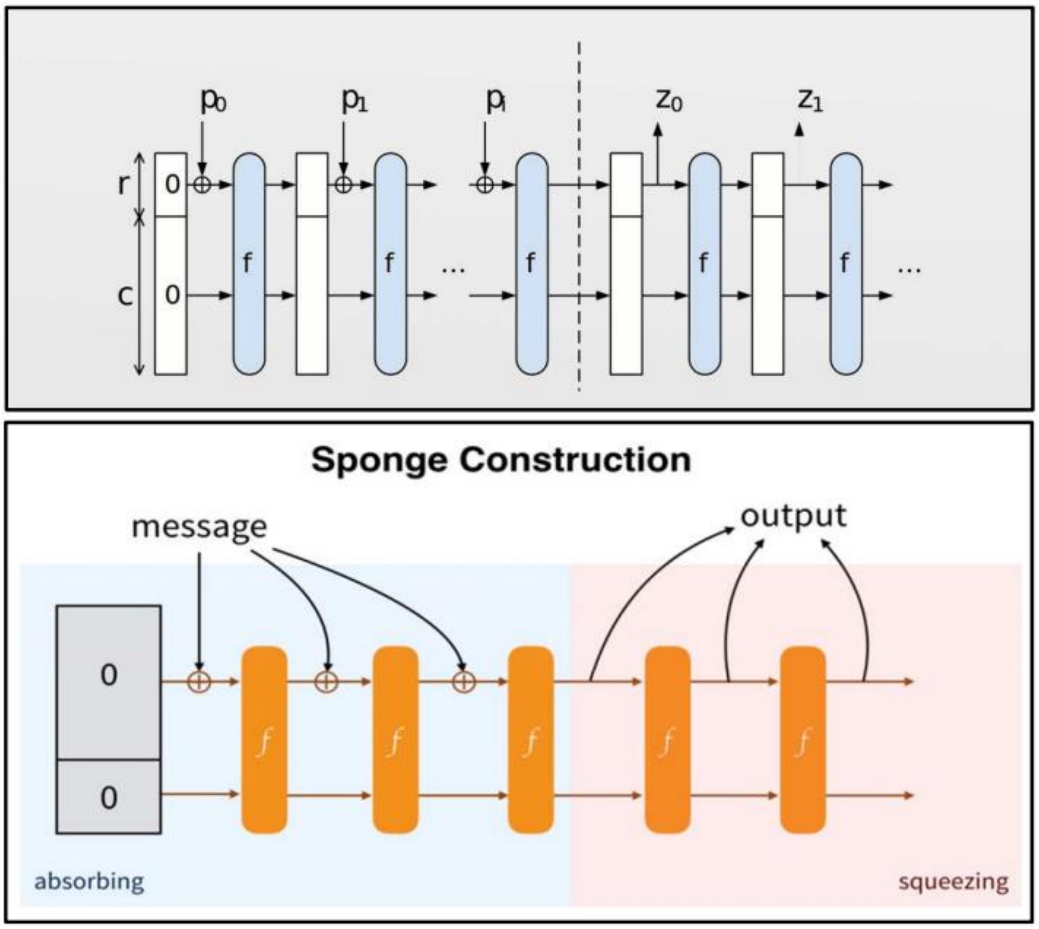


Figure (2.12) Block diagram of the SHA-3. [95]

2.12 Randomness Measurements

In cryptographic applications, there is a need to generate pseudorandom and random numbers. For example, the keys used for the encryption algorithm need to be random numbers. There is also a need to use random inputs in encryption protocols at different points. Such as those used for auxiliary quantities used to create digital signatures, or to create challenges in authentication protocols. There are a number of different statistical tests that are applied to a sequence to attempt to compare and evaluate the sequence with a truly random sequence. The probability property can describe the properties of a random sequence. Since there are possible outcomes of statistical tests, they are defined previously and can be described in probabilistic terms. There are a large number of statistical possibilities, each of which assesses the presence or absence of a "pattern" which, if detected, will indicate that the sequence is not random. Because the tests are numerous, it is not possible to consider a particular set of tests to be complete. Moreover, statistical test results should be interpreted cautiously to avoid incorrect conclusions about a particular generator [98].

2.12.1 NIST Tests

National Institute of Standards and Technology is a statistical package consisting of 15 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software-based cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests [98]. The 15 tests are:

1. The Frequency (Mono bit) Test,
2. Frequency Test within a Block,
3. The Runs Test,
4. Tests for the Longest-Run-of-Ones in a Block,
5. The Binary Matrix Rank Test,
6. The Discrete Fourier Transform (Spectral) Test,
7. The Non-overlapping Template Matching Test,
8. The Overlapping Template Matching Test,
9. Maurer's "Universal Statistical" Test,
10. The Linear Complexity Test,
11. The Serial Test,

12. The Approximate Entropy Test,
13. The Cumulative Sums (Cu sums) Test,
14. The Random Excursions Test, and
15. The Random Excursions Variant Test.

2.12.2 Entropy

Entropy is a measure of the randomness or chaos and unpredictability of the state in a system. A high degree of entropy indicates a chaotic and random series, while a low degree indicates a highly regularity of the system. The entropy H(X) of a random variable X is a measure of its average uncertainty. It is the minimum number of bits required on the average to describe the value x of the random variable X. Shannon Entropy [99]. The Shannon entropy of a random variable X is :

$$H(X) = -\sum_{x \in \mathcal{X}} P_X(x) \log P_X(x). \qquad (2.5)$$

In image encryption, if the information entropy values of the encrypted images are close to the theoretical value of 8, this indicates that it is difficult to perform a successful attack. The information entropy values change very slightly each time, because the random numbers that are entered into the image in each encryption[100][101].

2.13 Mean Square Error

Mean squared error (MSE) measures the amount of error in statistical models. It assesses the average squared difference between the observed and predicted values. When a model has no error, the MSE equals zero. As model error increases, its value increases. The mean squared error is also known as the mean squared deviation (MSD). A model with less error produces more precise predictions [102]. The formula for MSE is the following :

$$\mathrm{MSE} = \frac{1}{N}\sum\sum (E_{ij} - o_{ij})^2$$

Where, N is the image size, O is the original image, whereas, E is the edge image.

A. Motivation for Use as an Image Quality Metric

The mean squared error (MSE) for practical purposes allows comparing the “true” pixel values of the original image to the degraded image. The MSE represents the average of the squares of the "errors" between the actual image and noisy image. The error is the amount by which the values of the original image differ from the degraded image. The MSE represents the cumulative squared error between the compressed and the original image [102].

2.14 Peak Signal-to-Noise Ratio

The PSNR block computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image [102].

$$PSNR = 20\log_{10}\left(\frac{MAX_f}{\sqrt{MSE}}\right)$$

Where MAX_f is the maximum signal value that exists in our original “known to be good” image, MSE (Mean Squared Error).

2.15 Signal-to-Noise Ratio

SNR or signal-to-noise ratio is the ratio between the desired information or the power of a signal and the undesired signal or the power of the background noise. SNR describes the total noise present in the output edge detected in an image, in comparison to the noise in the original signal level. SNR is a quality metric and presents a rough calculation of the possibility of false switching; it serves as a mean to compare the relative performance of different implementations [102].

$$\text{SNR} = \left[\frac{\Sigma_i \Sigma_j (E_{ij})^2}{\Sigma_i \Sigma_j (E_{ij} - o_{ij})^2}\right]$$

Chapter Three

The Proposed System Design

Chapter Three
The Proposed System Design

3.1 Introduction

In this chapter, the security mechanism used to design the proposed encryption system, algorithms and work steps will be explained. The mechanism adopted to provide security for transmitted data depends on encryption and authentication. The adopted security mechanism was built and designed using lightweight encryption algorithms, with modifications made to them, and the use of the Chaos system to generate keys to give the algorithm more efficient encryption.

This system can provide a high degree of confidentiality for the data transmitted through IoT sensors to the server. It also provides the possibility of not reading the data even after it has been obtained by hackers. This system is designed to be effective in covering confidentiality, integrity and the inability to repudiate the various types of data transmitted through the different sensors. Providing protection for important data, such as company data, and medical or personal data, and encrypting it to ensure that it is not modified and stolen by attackers.

There is great flexibility in dealing with encrypted data for this system and great speed in encryption and decryption. In addition, the system allows checking message packages and their integrity upon receipt. The availability of such a system makes the user reassuring when transferring his data, ensuring that it does not fall into the wrong hands and ensuring its confidentiality.

3.2 The Proposed System Design

One of the main goals of the proposed system is to achieve the security and confidentiality of sensor data transmitted over the Internet of Things network. The proposed system consists of connecting three sensors (flame sensor, motion sensor and temperature and humidity sensor) through the Raspberry Pi attached to the surveillance camera. The camera transmits a live broadcast of the place permanently. If there is something abnormal, such as detecting moving objects or a fire, the camera will record a video of the place. The proposed lightweight and efficient encryption algorithm implementation, as well as the hashing function, is applied to the encoded video (frames). The encoded video is sent to the server, which decodes the video, takes the same hash function and compares it with the hash function sent to prove the integrity of the video.

Figure (3.1) shows the structure and relationship between the proposed system parts (the Proposed Lightweight Video Streaming Security Mechanism (PLVSSM), and other layer components. The presented model consists of three main stages:

1) Collection of data from the sensors (a set of sensors and a connected camera) to a Raspberry Pi which collects and controls the data.

2) Use the proposed security mechanism to protect this data, as the data is encrypted in Raspberry pi. The hash authentication is then used on the encrypted data to ensure that it is not tampered with on transmission.

3) Finally, this data is sent to the server to open it and do the necessary work depending on the type of problem.

In the data collection stage, there are 3 sensors connected to the Raspberry Pi as well as a camera capable of broadcasting live video of the place as well as capturing and storing images on the Raspberry Pi. The second stage is the stage of providing a security mechanism for sensor data and camera data before sending it. The proposed Lightweight Video Stream Security Mechanism (PLVSSM) consists of three stages: Chaos security key generation using Chaos system, Data encryption using Lightweight hybrid algorithms, and Authentication. Thus, PLVSSM consists of more than one algorithm combined into one security system, and these systems are As follows:

1. 9D Chaos Keys Generation (using a combination of 5D and 4D chaotic systems).
2. Hybrid Secure Hash Function (SHA3-256).
3. FPHLE: First Proposed Hybrid Lightweight Encryption Algorithm (Speck- Hummingbird Algorithm).
4. SPHLE: Second Proposed Hybrid Lightweight Encryption Algorithm (Hummingbird- Speck Algorithm).
5. TPHLE: Third Hybrid Lightweight Encryption Algorithm (The Two algorithms work together in parallel).

Data blocks are encrypted using hybrid cipher algorithms, which consists of more than one algorithm (speck and hummingbird). Where the algorithms are combined in a different way each time, which is to take the basic parts of the first algorithm and insert them as layers in the second algorithm. This method helps get rid of weak parts of the algorithm that may cause poor performance. The generated Chaos keys will be used for encryption, as well as hashing, in all PLVSSM algorithms.

Hybrid SHA-3 authentication was used on the encrypted data generated by the algorithm to further complicate hashing and protect sending videos from tampering. The Chaos Keys Generation algorithm was used to generate random numbers using a combination of the 5D and 4D Chaotic systems with initials and various parameter values to produce 9-D Chaos Key values. Chaos keys used in all PLVSSM algorithms: in the generation of cipher/hash keys, and in some cipher/hash functions.

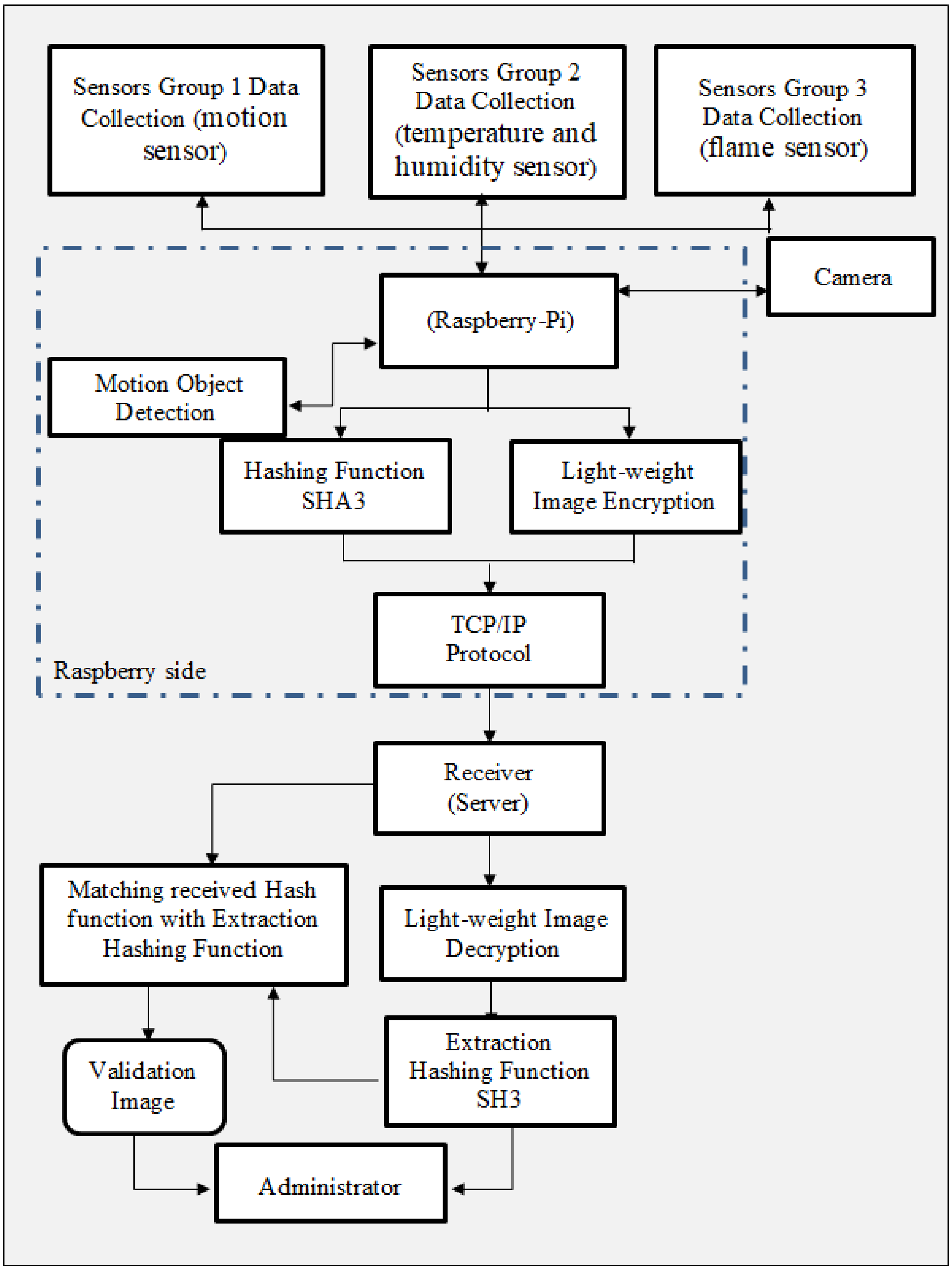

Sensors Group 1 Data Collection (motion sensor)
Sensors Group 2 Data Collection (temperature and humidity sensor)
Sensors Group 3 Data Collection (flame sensor)
Camera
(Raspberry-Pi)
Motion Object Detection
Hashing Function SHA3
Light-weight Image Encryption
TCP/IP Protocol
Raspberry side
Receiver (Server)
Matching received Hash function with Extraction Hashing Function
Light-weight Image Decryption
Extraction Hashing Function SH3
Validation Image
Administrator

Figure (3.1): General Diagram of the Proposed System Structure

3.2.1 Sensing Data Aggregation layer

Data is transmitted and received through the system in a wired manner. The confidentiality and security of this data must be maintained by applying encryption algorithms on the sensor data in the Raspberry pi. The data is collected by the sensors and the camera from the environment in which they are located and stored within the Raspberry pi. Sensors obtain data from the environment in many different ways. The symbol D sensor indicates data incoming from a sensor node attached to the Raspberry Pi controller.

The symbol R_{base} also stands for the wired connection between the Raspberry pi device and the sensors. There is a connection between the Raspberry pi device and the server (the responsible party) denoted by the R_{server}, which processes the data after receiving it. R_{base} uses the PLVSSM algorithm to encrypt sensor data, while R_{server} validates and authenticates the packet and sends the authorized packet to the administrator's machine to decrypt the data. Algorithm (3.1) demonstrates data collection from sensors.

Algorithm (3.1) Sensing Data Aggregation

```
Input: (Sensors, Raspberry pi $R_{server}$, Raspberry pi $R_{base}$, network parameters,
and connections links).
Output: Sensing data

Begin
Step 1: While the $R_{server}$ and $R_{base}$ state= start
Step 2: For all Sensors groups: Initial the sensors, Raspberry Pi, network
parameters, protocol, environment, and connections links.
Step 3: Get state for each sensor in group connect to $R_{base}$.
Step 4: If any sensor state=stop send group failure signal to administrator
```

through R_{server}.

Step 5: Collect and aggregates sensing data D sensing ={ $D_{sensor1}$, $D_{sensor2}$, … $D_{sensor\ n}$}

Step 6: Pass sensing data to the PLVSSM algorithm.
Step 7: Repeat steps 4 and 5 until R_{server} or R_{base} state= released
End.

3.3 The Proposed Motion Object detection module

Motion is detected in the video if the difference between the current and the previous frame is not zero. When movement occurs, the pixel values differ from one frame to another, so it is possible to determine the movement of the object in a particular frame. After capturing the video, it is sent to the server after it is encrypted and authentication is applied. With this processing, the captured and sent image will depend on the event and not periodically.

Algorithm 3.2: Motion Detection Algorithm

Input: Cam- Frames

Output: Assign Masks //represented motion Object Mask

```
Begin
  Steps
     1: Ref ←Reading the background image // input the background image
     2: N← Calculate the number of frames // find the number of frames per second
     3: $N_{Row}$←find number of rows in background // find row in background size
     4: $N_{col}$← find number of columns // find column in background size
     5: For k←1 to N
     6:     $Fr_{Current}$←CamFrames(1to$N_{Row}$,1 to $N_{col}$, k) //specifying current frame
     7:     For i ← 1 to $N_{Row}$
     8:         For j← 1 to $N_{col}$
     9:             if abs(Frcurren(i,j)-Fback(I,j))>threshold1 // finding the differences of current
```

```
                                        frame with respect to background
    10:              Mask(i,j)←1 // assigning mask to one if there is a difference
    11:                Sum=sum+1 // count number of white pixels in mask
    12:          Else
    13:              Mask(j,j)←0   // assign the value of mask to zero
    14:        end j
    15:      end i
    16:   if sum>Threshold2  // comparing motion with threshold to find motion object
    17:      assign Motion on Cam-frames
    18:     end if
    19: end for k
End Algorithm
```

3.4 Proposed IoT Video Streaming Lightweight Security Mechanism (PLVSSM)

Execution and processing time is one of the biggest challenges in designing security systems for IoT applications. Especially if the use of more than one algorithm is relied upon under the concept of hybrid algorithms. PLVSSM is designed with minimal resource and storage space requirements while minimizing the number of rounds to reduce encryption time. Since video encryption requires high processing and encryption speed, the system with rich features contributes to providing security with efficient implementation very quickly. The proposed lightweight security mechanism for an IoT system contains three stages: chaos keys generation using chaotic system, data encryption using the proposed (Speck, Hummingbird) algorithms, and data hashing using SHA-3. Since Raspberry pi is where the data is collected, the data in it must also be encrypted.

3.4.1 The 9-D Chaos Keys Generation Stage

The chaotic system is included in the work to increase safety and strength, due to the random properties of the output numbers of the chaotic systems. Where chaotic keys are widely used in encryption operations. This is by taking advantage of the property of large change that occurs in later cases once there is a small change in the initial stages of a nonlinear system. Because a random key is needed in PLVSSM algorithms for their security processes to use messy 9-D (K1, K2, ..., K9) keys. The 9-D messy key generation system is designed and proposed in the key generation process. This system is a system consists of a four-dimensional Chaotic System (from [103]) and another five-dimensional Chaotic System (from [104]). And that's with different initials and parameter values to create 9-D clutter key values. This system was used in the Chaos Key generation stage for proposed hybrid algorithms.

$$\left.\begin{array}{l} xt[i+1] = xt[i] - a*(1-xt[i]-zt[i])*dt \\ yt[i+1] = yt[i] + (-yt[i]-xt[i]*zt[i]+r*yt[i]2)*dt \\ zt[i+1] = zt[i] + (xt[i]*yt[i]-b*zt[i])*dt \\ kt[i+1]=kt[i]+(c * yt[i] * (xt[i]*zt[i] - kt[i]))*dt \end{array}\right\} \quad \ldots. (3.1)$$

Where a, b, c, r is the chaos parameters. While xt,yt,zt, kt and is the initial conditions for chaos map.

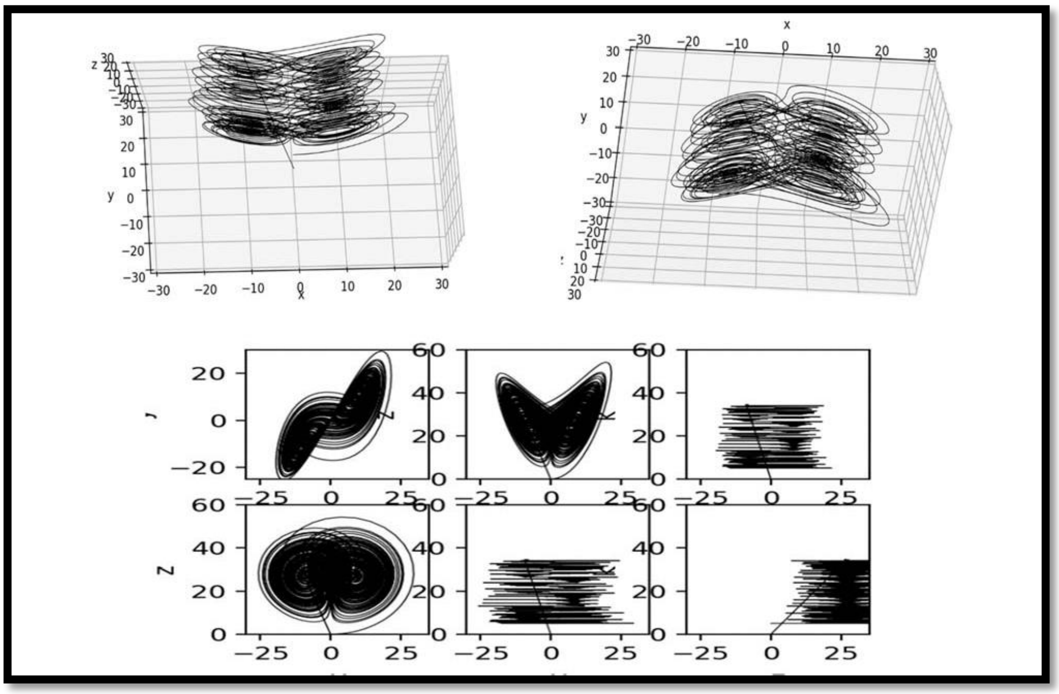


Figure (3.2): Map of the First 5-D Chaotic System.

The second five-dimensional chaotic system is the second part of the chaos generator. Equation (3.2) shows the equations of the chaos system and its parameters.

$$\left.\begin{array}{l} x = x +(-s*x+ y*k-r*p) \\ y= y +(-y-x*z + r*x-u*p) \\ z= z +(z*x*y-1.5*s*p-k) \\ k= k+(s * x + (u*y-r*k)) \\ p=p+(b*((x + k)/z)+y) \end{array}\right\} \qquad (3.2)$$

where x, y, z, k, and p are the vectors that represent the behavior of the system. The behavior shows chaotic system for s =0.95, r = 0.5, b=0.01, u =1.1. The significance of the proposed system is that it produces a new 4D chaotic system that is more complex and sensitive to initial conditions. In this system, five-dimensional system results with five positive Lyapunov exponents which are 0.0199, 0.017, 1.496, 0.015, and 0.016 as the order of the system maps.

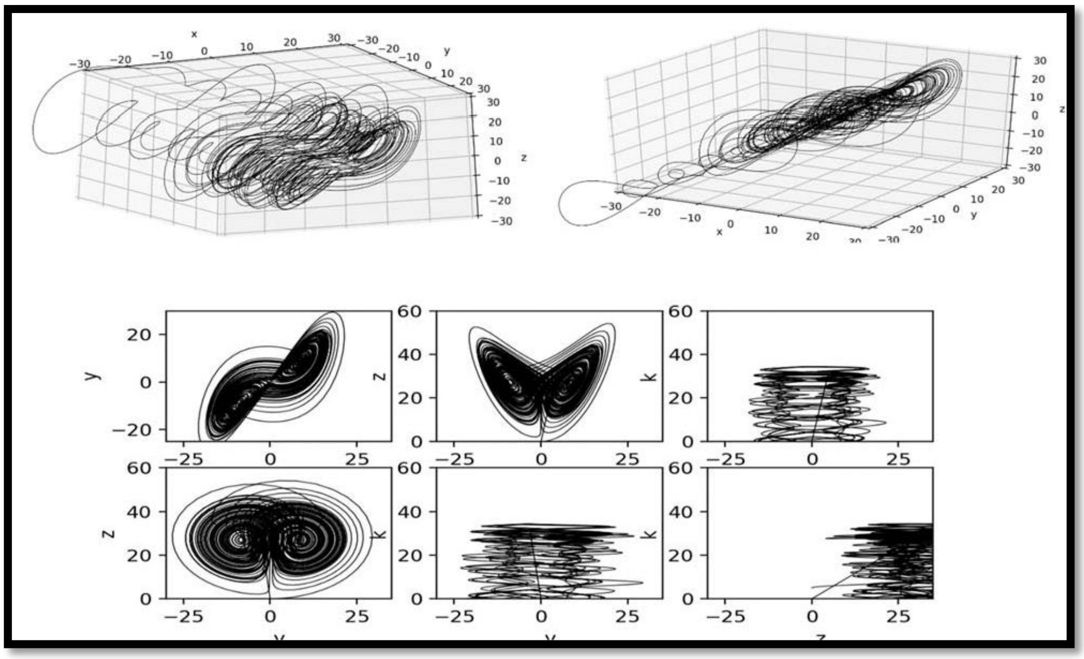


Figure (3.3) The Map results of the second 5-D chaotic system.

The figure (3.4), it is shown Parameters and initial value for proposed chaos system generator.

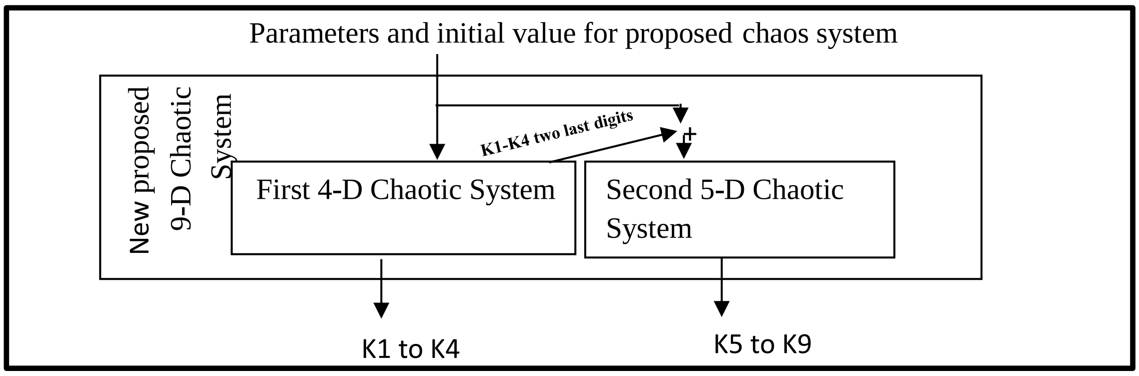

Parameters and initial value for proposed chaos system
New proposed 9-D Chaotic System
K1-K4 two last digits
+
First 4-D Chaotic System
Second 5-D Chaotic System
K1 to K4
K5 to K9

Figure (3.4): The block diagram of the propose Combination Chaotic Generator.

The nine random keys generated (K1, K2, K3, K4, K5, K6, K7 and K9) are used in all PLVSSM operations. The algorithm (3.3) illustrates the process of generating chaos keys.

Algorithm (3.3) key Generation by the Chaos.

```
Input: Initial parameters and values.
Output: Chaos keys results (K1, K2 …, K9).
 Begin
Step 1: Loading the initial values and parameters of the chaotic system suggested.
Step 2: Apply equation (3.1) to generate the first 4-D chaos keys (K1 to K4).
Step 3: Load the parameters of the new proposed chaotic system.
Step 4: Get the last significant value from the key Generation (K1 to K4).
Step 5: Add the split significant values to the selected initials values of the second part's proposed chaotic system dimensions.
Step 6: Apply the equation (3.2) to generate the last part of chaos keys (K5 to K9).
Step 7: Save the chaotic keys created in the hexadecimal number format file.
End.
```

3.4.2 The Encryption Stage in PLVSSM

At this point, the proposed lightweight encryption algorithms (speck and hummingbird) were used to provide security for camera data. The first step in the Internet of Things system is to collect data from sensors and apparatuses communed to the Internet of Things system. In this proposed work, three sensors were utilized in addition to a camera and they are controlled by a Raspberry Pi 4B. The second step is to provide security and encryption for the video' data before sending it to the server. This is done by applying the proposed safety mechanism, which is the hybrid algorithm (speck Hummingbird). Figure (3.5) shows a flowchart of the proposed encryption algorithms, in terms of random key generation and authentication use. As shown, encryption is only done when a movement appears.

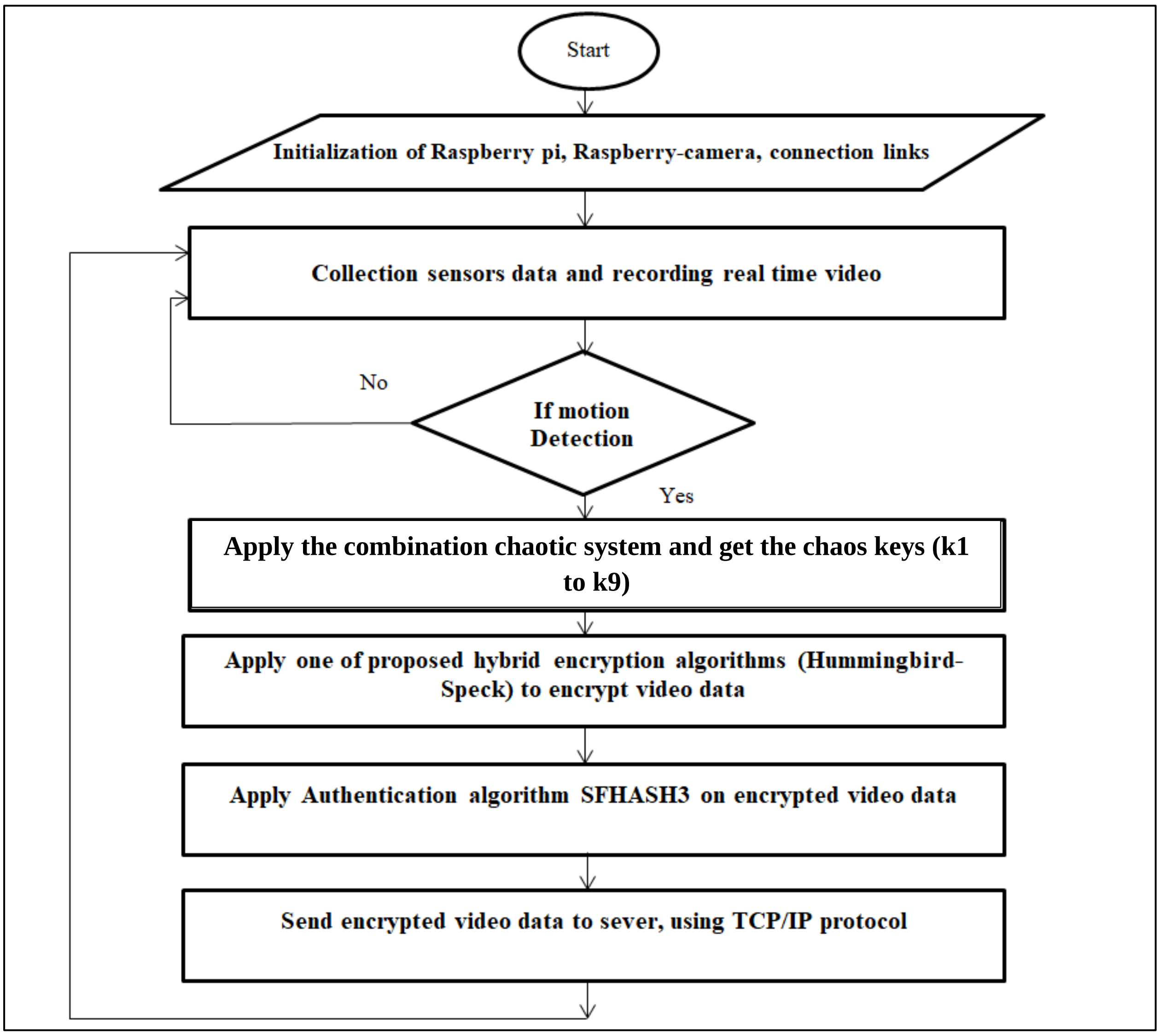

Start
Initialization of Raspberry pi, Raspberry-camera, connection links
Collection sensors data and recording real time video
No
If motion Detection
Yes
Apply the combination chaotic system and get the chaos keys (k1 to k9)
Apply one of proposed hybrid encryption algorithms (Hummingbird-Speck) to encrypt video data
Apply Authentication algorithm SFHASH3 on encrypted video data
Send encrypted video data to sever, using TCP/IP protocol

Figure (3.5) The Flowchart of Applying the Proposed Hybrid Encryption Algorithms (Sender Side).

.

3.4.2.1 First Proposed Hybrid Lightweight Encryption Algorithm (FPHLE)

The **FPHLE algorithm** was designed by merging the Speck algorithm (one round) with the Hummingbird algorithm. Figure (3.6) illustrates a block diagram of the Hybrid Hummingbird -Speck Algorithm. The Speck algorithm was inserted as a layer in the Hummingbird round layers to increase the complexity of the encrypting results. The suggested hybridization the Hummingbird algorithm with the Speck algorithm in order to increase randomness strength of the whole Hummingbird structure processing, even if its key generation is weak.

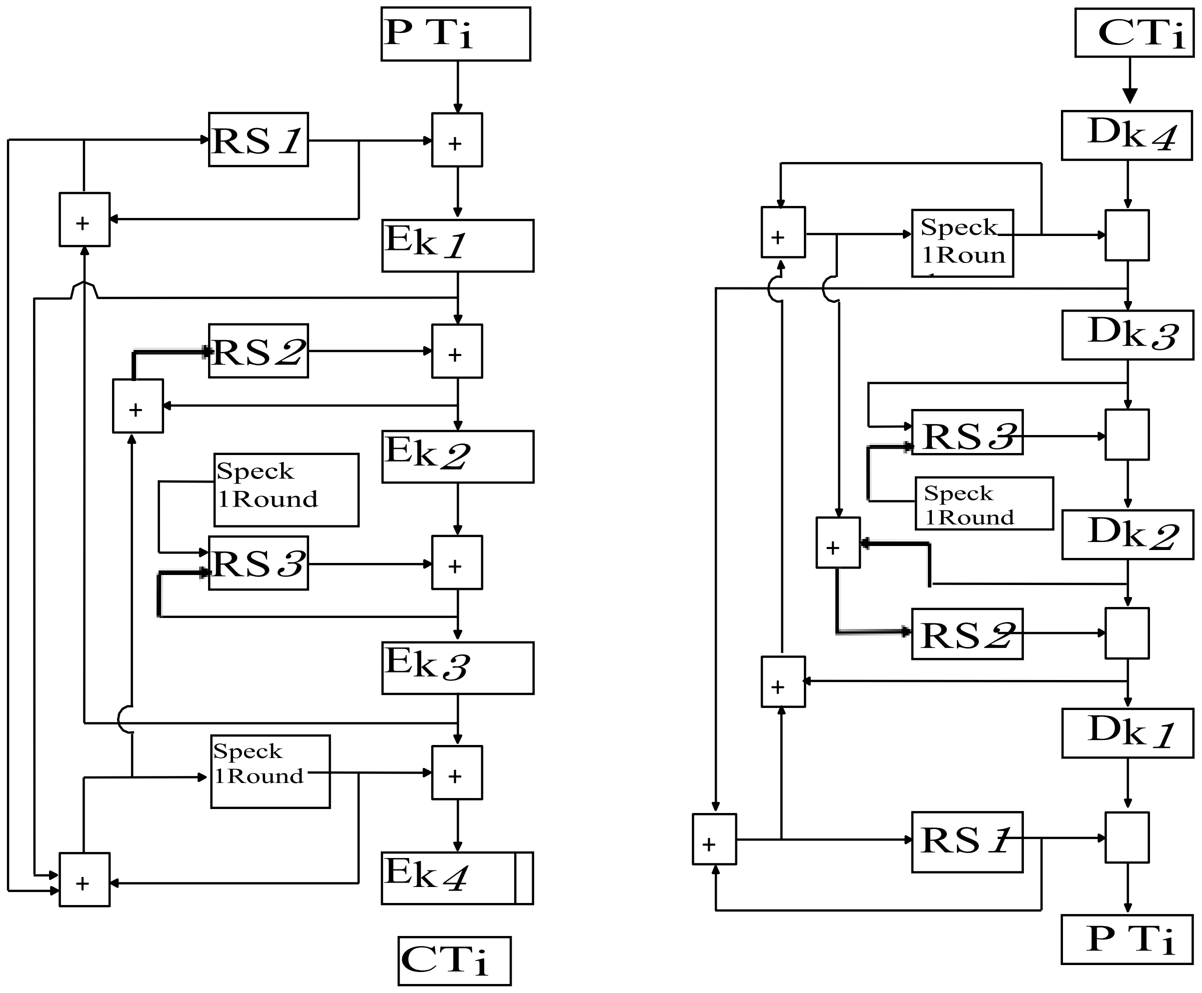
After collecting the video frames data from the data collection phase in the form of image frames streaming file, it enters to FPHLE algorithm, where the following stages of FPHLE work:

1) Divide this file data into blocks from 128 bits.
2) Using a Chaos key generator to generate strong and secure random numbers.
3) The total number of rounds for the hybrid algorithm is 16 rounds, which hybrid algorithm rounds include the XOR operation, the S-box which consists of 4 S-boxes different parts, Speck algorithm with one round.
4) After completing 16 rounds from hybrid algorithm rounds in Hummingbird structure with the 256-bit key with four inner registers to 80-bit internal state to produces 64-bit encrypted data.
5) The Speck algorithm is used twice in this proposed. The enter encrypted data to Speck in first step using NONCE and K2, while the second using of the Speck algorithm is used to encrypted $RS4_t$ using K3 in

Hummingbird encryption/decryption processing for produce final encrypted data in file.

6) The chaos keys (K1,…, K9) used in the different stages of encryption/decryption processing steps for the purpose of generating strong and secure random numbers for a structure of Speck.

These stages are in the encryption, but in the decryption, is reversed. Where the following algorithms (3.4-3.5) illustrate the steps for the FPHLE.



(a) Proposed Encryption Processes (b) Proposed Decryption Processes

Figure (3.6) The First Proposed Lightweight Encryption Algorithm

Algorithm (3.4): The First Proposed Hybrid Lightweight Encryption.

```
Input: A 16-bit plaintext $PT_i$ and four rotors $RSi_t$ $(i = 1; 2; 3; 4)$
Output: A 16-bit ciphertext $CT_i$
1: $V\ 12_t = Ek1\ (PTi \boxplus RS1_t)$          [Block Encryption]
2: $V\ 23_t = Ek2\ (V12_t \boxplus RS2_t)$
3: $V\ 34t = Ek3\ (V23_t \boxplus RS3_t)$
4: $CT_i = Ek4\ (V34_t \boxplus RS4t)$
5: $RS3_t$=Speck (NONCE, K2) one round.
6: $RS1_{t+1} = RS1t \boxplus V\ 34t$
7: $RS3_{t+1} = RS3_t \boxplus V\ 23t \boxplus LFSR_{t+1}$
8: $RS4_{t+1} = RS4_t \boxplus V\ 12t \boxplus RS1_{t+1} \boxplus$ Speck ($RS4_t$, K3) one round.
9: $RS2_{t+1} = RS2_t \boxplus V\ 12t \boxplus RS4_{t+1}$
10: return $CT_i$
```

Algorithm (3.5): The First Proposed Hybrid Lightweight Decryption.

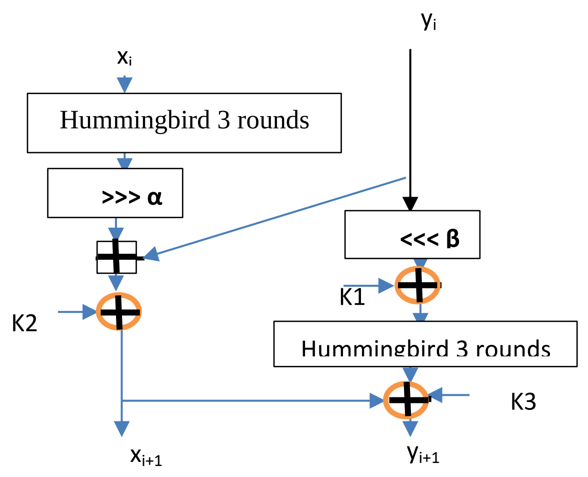
```
Input: A 16-bit ciphertext $CT_i$ and four rotors $RSi_t$ $(i = 1; 2; 3; 4)$
Output: A 16-bit plaintext $PT_i$
1: $V34_t = D_{k4}\ (CT_i) \boxminus RS4_t$          [Block Decryption]
2: $V23_t = D_{k3}\ (V34_t) \boxminus RS3_t$
3: $V12_t = D_{k2}\ (V23_t) \boxminus RS2_t$
4: $PT_i = Dk1\ (V12_t) \boxminus RS1_t$
5: $RS3_t$=Speck (NONCE, K2) one round.
6: $RS1_{t+1} = RS1_t \boxminus V34_t$
7: $RS3_{t+1} = RS3_t \boxminus V23t \boxminus LFSR_{t+1}$
8: $RS4_{t+1} = RS4_t \boxminus V12_t \boxminus RS1_{t+1} \boxminus$ Speck ($RS4_t$, K3) one round.
9: $RS2_{t+1} = RS2_t \boxminus V12t \boxminus RS4_{t+1}$
10: return $PT_i$
```

The Variables that used in Algorithms (3.4,3.5) explained and described in Table (2.7) in chapter two.

3.4.2.2 The Second Proposed Hybrid Lightweight Encryption (SPHLE) Algorithm

SPHLE algorithm was illustrated in figure (3.7) as a second proposal used in the proposed system to encrypts the frames of video. The video is encrypted using a proposed algorithm by segmenting the frames into blocks per 128-bit block. Speck algorithm is hybrid with Hummingbird algorithm and applied for ten rounds. These steps are repeated for ten rounds and produce the output for sending it to the server.

The proposed encryption algorithm is a hybrid Speck-Hummingbird that is designed from combined the Speck algorithm (with 10 rounds to reduce the Speck encryption time) with the Hummingbird algorithm (with 3 rounds). The Hummingbird algorithm was inserted as a layer in the Speck round layers to increase the complexity of the encryption results and to avoid many attacks. The generated keys were distributed between the Hummingbird algorithm and Speck algorithm as encryption keys. In this case, the key schedule was removed from the Speck algorithm and replaced by generated chaos keys. These chaos keys were used to raise the randomness of the output cipher text and give the best strengths to the encryption algorithm. Algorithm (3.6) illustrated the steps of SPHLE.



Figure(3.7) The Second Proposed Hybrid Lightweight Encryption Process

Algorithm (3.6) The Second Proposed Hybrid Lightweight Encryption

```
Input: data block, chaos key(K1,…, K9), $\alpha$, $\beta$ , rounds
Output: cipher block.
Begin
 1: Load input block data of size 128 bits into two parts of 64 bits ($x_i$ and $y_i$).
2: load chaos keys with size 256bit.
3: for i=0 to rounds
    3.1: Hummingbird ($y_i$ , [k1,k2,k4,k9])
    3.2: $x_i = x_i >>> \alpha$, $y_i = y_i <<< \beta$
    3.3: $x_i = x_i + y_i$ ,  $y_i = y_i$ xor K1
    3.4: $x_{i+1} = x_i$ xor K2
    3.5: $y_i$= Hummingbird ($y_i$ , [k5,k6,k7,k8])
    3.6: $y_{i+1} = y_i$ xor K3
    3.7: swap between $x_{i+1}$ and $y_{i+1}$, then return $x_{i+1}$, $y_{i+1}$
4:      Return final encrypt x, y
5: end
```

3.4.2.3 The Third Proposed Hybrid Lightweight Encryption (TPHLE)

The TPHLE designed also by merging the Hummingbird algorithm (three rounds) with the SPECK algorithm (one round). Figure (3.8) illustrating a block diagram of the Hybrid SPECK-Hummingbird Algorithm. The TPHLE is designed in type of Feistel structure type with 10 rounds. The SPECK algorithm was taking the left side of the proposed TPHLE, while the Hummingbird take the right side. In each end of the round, the left and right sides will be swapped. The round designed to raise the complexness of the data encryption outcomes.

The collected sensory data will be passed to TPHLE, where the following stages below of TPHLE works:

1) Dividing file of video data into blocks from 128 bits, then divide 128 bits into 64 bits for both sides (left and right).
2) Using Chaos key generator to generate nine random keys strong and secure. These nine keys will be used in Hummingbird and Speck algorithms.
3) The total number of rounds for the hybrid algorithm is 10 rounds, which hybrid algorithm rounds include the two algorithms (Hummingbird and Speck) work in the same time (parallel), XOR operation, modular add operation, and circular shift operations on both sides.
4) The output of the two sides will be shifted (by 8 cycles right for right side, or by 8 cycles left for left side). Then swap operation between the two sides will be done.

The algorithm (3.7) illustrates the steps of TPHLE encryption operation.

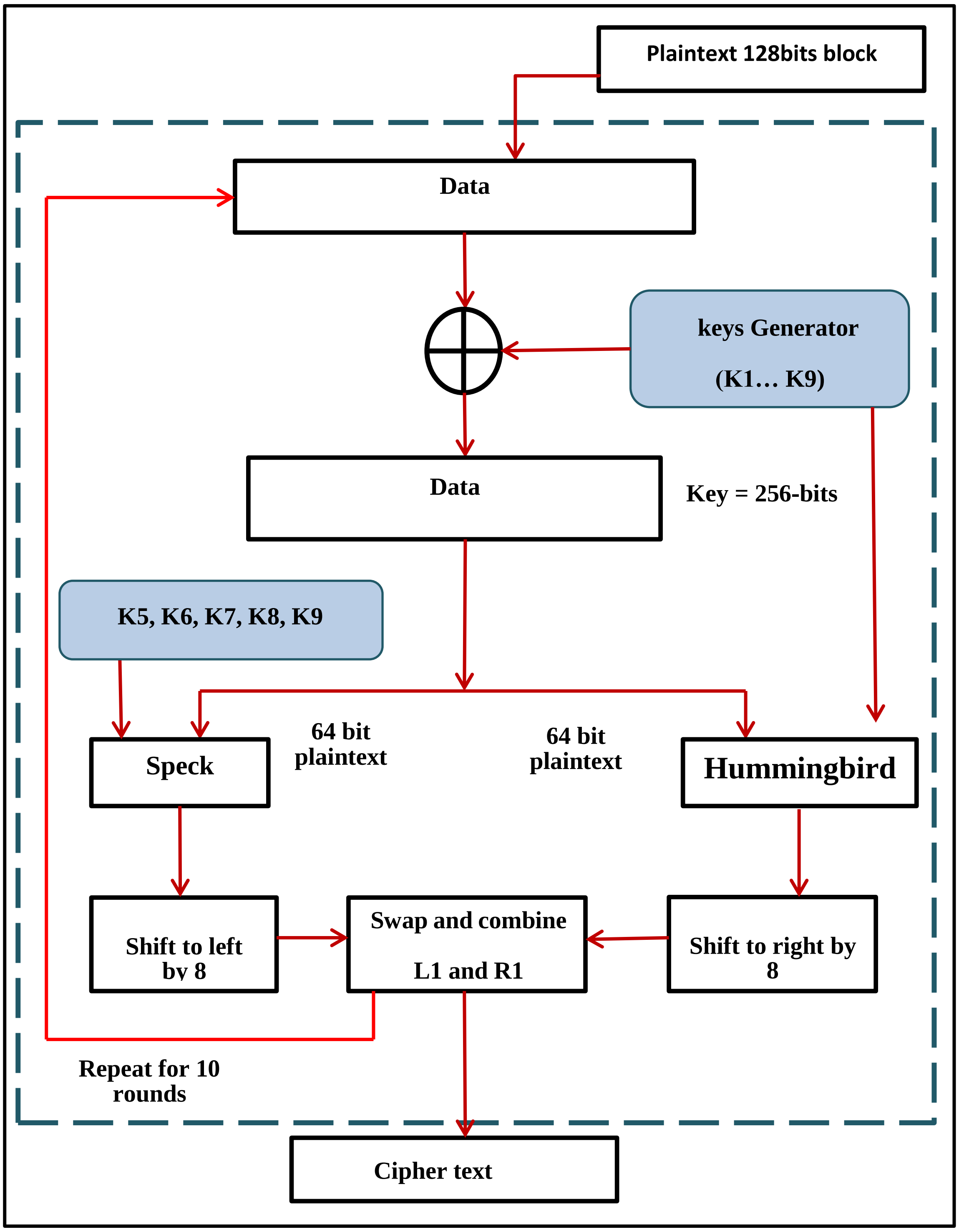


Figure (3.8) Block Diagram of the TPHLE

Algorithm (3.7) The Third Hybrid Lightweight Encryption (TPHLE) Algorithm

Input: Sensor data, key generator1, key generator2.

Output: Encrypted data.

```
Begin:
Step1:  Divide input data into   blocks of 128 bits.
Step 2: Use key generator 1 and divide the 256-bit secret key into nine parts
        of the sub keys (K1, K2, ......, K9) each part is 32 bits.
Step 3: Apply the XOR operation between the input data and keys (K1,.., K9).
Step 4: Split the outputted data from step3 into the left (L) and right (R) sides by 64 bits.
Step5:  Apply Hummingbird with 3 rounds on the right-side data (R), and Speck with one round on the data of left-side (L).
Step6:  Apply the shift operation on the right side by 8 to right, and left-side data by 8 to left.
Step 7: Swap between data in right and left sides.
Step8: repeat steps from step 2 to 7 until complete 10 rounds.
Step9: Stop the process permanently when getting to the completion of all
        sensor data.
End
```

3.4.3 Proposed Authentication Layer in PLVSSM

The SHA-3 hash was used in the proposed security mechanism to authenticate the encrypting video data . SHA-3 runs on E-Stream technologies and uses two keys generated by the proposed combination chaotic system to replace the initial values used. At first, the data is encrypted by the proposed encryption method, and then the encrypted data is sent to SHA-3 for authentication and the creation of a serial hash for it. Figure (3.9) shows a diagram of the authentication method proposed in PLVSSM. Also it shows that the PLVSSM operations that include the authentication steps and all parameters and raw values entered are done on both sides, of the sender side (Raspberry pi sensors) and the recipient side (IoT server).

The received video is validated from the server side of the IoT by calculating the hash using the proposed lightweight authentication scheme on all received data. The received payload hash results are compared with the hash log of the payload sent by the user. Depending on the result of the comparison, the packet is accepted or rejected. On the server side, the final hash is generated by applying the same operations to the video data Raspberry pi. The results for the final hash from the server side are compared with the final hash received from the sensors packet associated with Raspberry pi. This applies to every packet received by the IoT server.

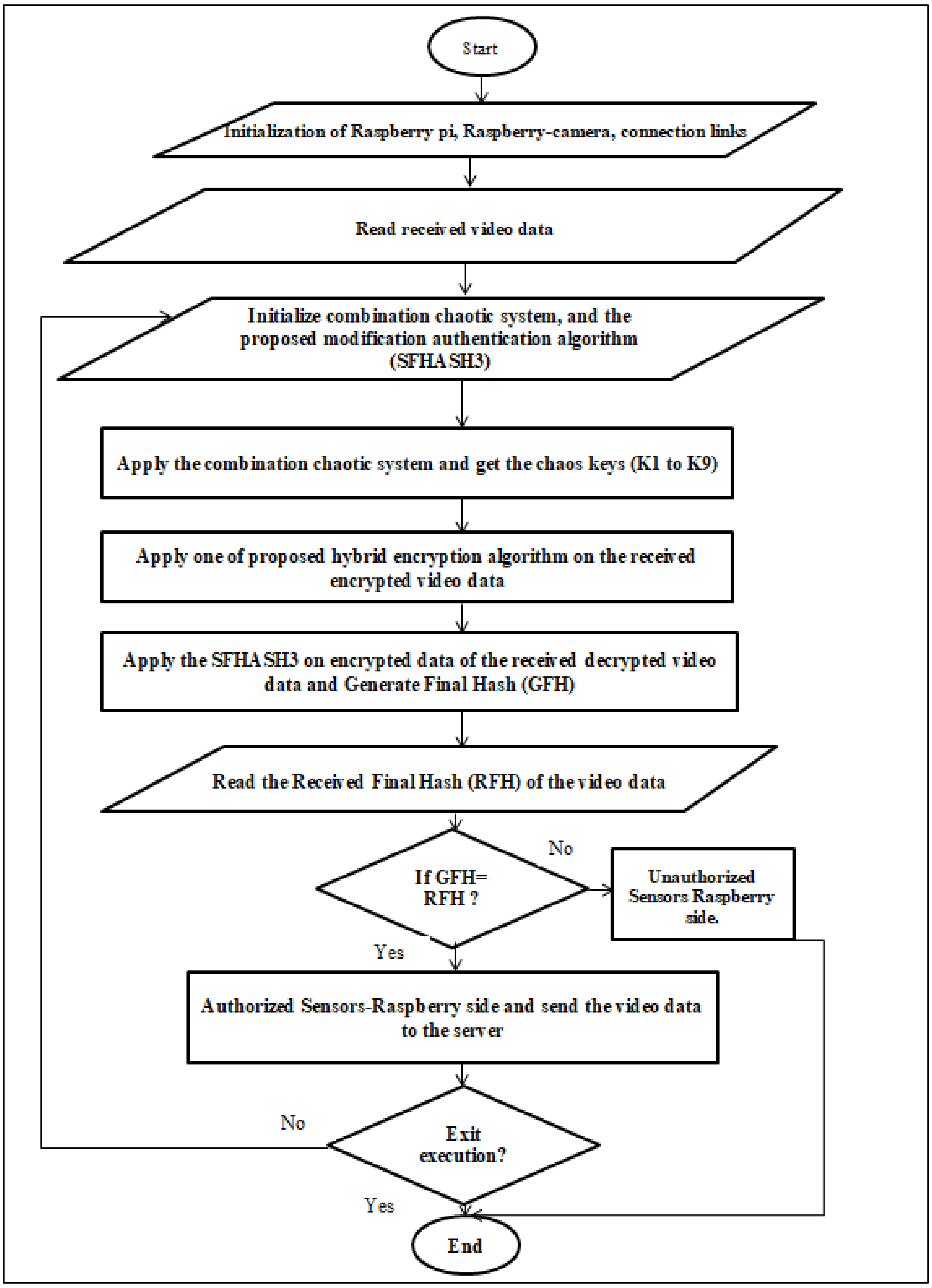

Start
Initialization of Raspberry pi, Raspberry-camera, connection links
Read received video data
Initialize combination chaotic system, and the proposed modification authentication algorithm (SFHASH3)
Apply the combination chaotic system and get the chaos keys (K1 to K9)
Apply one of proposed hybrid encryption algorithm on the received encrypted video data
Apply the SFHASH3 on encrypted data of the received decrypted video data and Generate Final Hash (GFH)
Read the Received Final Hash (RFH) of the video data
If GFH= RFH ?
No
Unauthorized Sensors Raspberry side.
Yes
Authorized Sensors-Raspberry side and send the video data to the server
No
Exit execution?
Yes
End

Figure (3.9) Proposed authentication algorithm in PLVSSM (server side)

A. Hybrid SHA3-256 bits algorithm

In this stage, certain the SHA3 by using the Hybrid SHA3-SPECK-FPHLE Algorithm (SFHASH3) in order to achieve the following:

1- Ensure that the message is correct (ensuring that the data is not tampered or modified).

2- This message has to be issued by an authorized authority.

There is a weakness in SHA3 in that it losses the required speed for achieving constant-length fragmentation. Which slows down the work when using SHA-3 for validating the information. So was proposed the design of SFHASH3 by merging the SPECK- FPHLE algorithm (with nine rounds for the purpose of speeding up the encryption process) as an additional layer complementing the sponge structure layers found in the SHA3. Figure (3.10) illustrates a block diagram of the hybrid SHA3-SPECK algorithm. SPECK algorithm was inserted as an additional layer to the sponge structure layers of SHA3 to increase the complexity of the algorithm as well as obtain a stronger hybrid SHA3 structure and different from the original SHA3 structure.

Where the following algorithm (3.8) illustrates the proposed SFHASH3 work. In addition, the basic idea of SFHASH3 that includes several steps is shown below:

1. Entering the plain file into the SPECK algorithm for the purpose of the encryption process, which includes dividing this file data into blocks of 128 bits, so that each side (right-left) contains 64 bits, and with the generate the key size 256-bit.

2. With rounds 9, in order to reduce encrypting process time and also to be suitable for restricted devices. Where each round consists of three main operations (modular add, XOR and circular shifts to left and right).
3. Merging the result of the encrypted data file from the SPECK algorithm with the encrypted text produced from Hummingbird &SPECK, then split them into k-blocks of fixed size. The resulting splits are a sequence of bit blocks (P0, P1, P2, ..., Pk - 1). These blocks are then passed to the proposed SFHASH3 hashing algorithm.
4. The SFHASH3 design will be the basis of the SHA-3 architecture. The sponge design is a simple repetitive construction where the blocks are absorbed, forming a padded input message, then whenever we require the output, the sponge is compressed to get the maximum possible output.
5. The sponge function is constructed from three components (a parameter called the rate, denoted by **r**, a basic function in fixed-length strings denoted by the symbol **f** and the padding base denoted by a pad), where the sponge function operates on the initial value **r**, which is the size of the block used divides the input data resulting from merging the encrypted data by the proposed algorithm and the original data, where this value is called bitrate.
6. Also, the sponge structure consists of 2 stages: the absorption stages that at this phase the input block to be processed is padded with zeros to boost its length from r bits to state bits. Then comes the squeezing stage in which the first bits are kept as Zi block and concatenated with function f that the previously generated blocks.

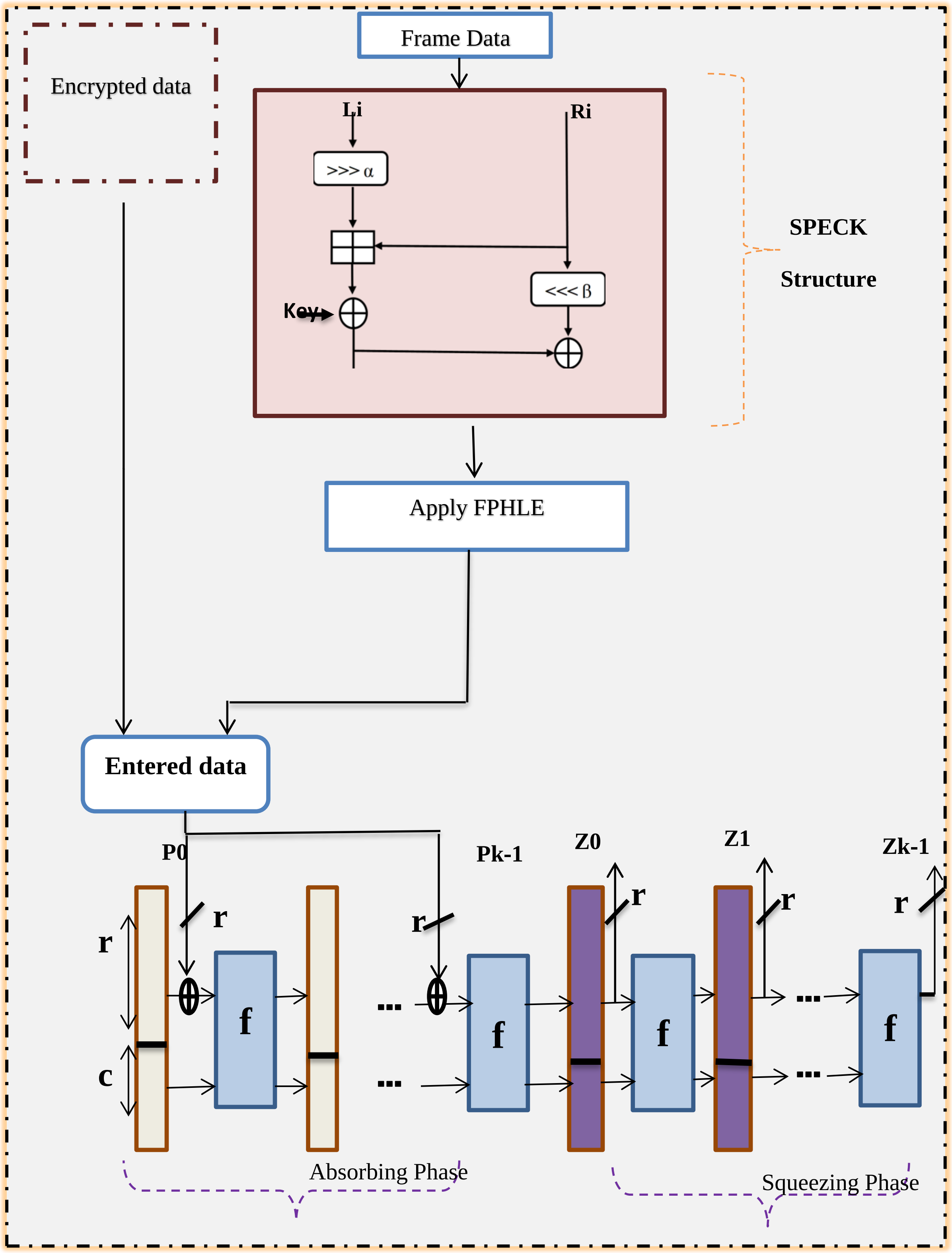


Figure (3.10) Block Diagram of SFHASH3

Algorithm (3.8) The Proposed SFHASH3

Input: Data content(file), k=256.

Output: Final Message Digest.

Step1: Start with input data and divide it into a block of 128 bits. Also, used key size 256 bit.

Step2: Merge encrypted text with the encrypted data produced from FPHLE &SPECK, and then read it and partition it into series of bit blocks (P0, P1, P2…., Pk–1).

Step3: Use the keys as the initial value in (R, C) the value handled on each iteration in SFHASH3.

Step4: Use the F-function of KECCAK to calculate the message digest using one iteration a function F.

Step5: Repeat previous process until reached completes rounds in order to produce
a new message digest (Z0).

Step6: Save message digest(Z0).

Step7: Return to step 4 to produce a new message digest (Zi), previous processes are repeated until all the contents of the input data are finished.

Step8: End

Chapter Four

Proposed System Implementation, Results and Discussion

4.1 Introduction

After the proposed system design is completed, the role of implementing this system is in the practical environment and seeing the results using the Raspberry pi device and associated sensors. The design of the execution system included the implementation of data encryption and decryption based on the generation of chaotic keys for the proposed algorithms. It also provided an authentication process for encrypted data to ensure that data and videos are not tampered with. In this chapter the results of the proposed system implementation will be clarified, analyzed and discussed, as well as the fact that the system has been tested with various security tests.

4.2 Proposed System Implement

In the third chapter, it was clarified that the proposed system consists of three layers: the data collection layer, the lightweight safety mechanism provision layer (PLVSSM), and the emergency response layer to create the appropriate response to the event. The first step in the system includes collecting information and data from the sensors associated with the Raspberry pi device. In this work, 3 sensors were used to test the security of the proposed system. After receiving the data by the responsible calculator, the encoded data is analyzed to generate an appropriate response.

4.3 Requirements and Environment

The proposed system was implemented in the presence of various requirements such as hardware requirements (Raspberry pi device, sensors, camera and wires to connect sensors). In addition to software requirements such as the operating system for the Raspberry pi device, which is Raspbian, the programming language Python 3.9, and a computer with the operating system (

Windows 10). Three Various types of sensors were used, a motion sensor, a sensor for measuring temperature and humidity, a sensor that senses the smell of gas produced by fires, in addition to a Raspberry pi camera. Table (4.1) shows the types of sensors used and a simple explanation of each.

Table 4.1: Sensors and their details

Sensor Type	**Description**	Used Sensor name
Raspberry pi camera	Used to sense the environment for any changes in the visual scene, the data represented in image form, image processing methods used to process the data of camera sensor.	Camera Pi 5 Megapixel
Humidity Sensor	Used to sense the changes in the Humidity scene, the representation of data is on numerical form, its output used in monitoring in sensor area.	DHT-11
Fire Flame Sensor	Used to monitor the environment for the change in the temperature and fire flame caused by fire, Flame sensor represents the data on numerical form and used to monitoring sensor area for firing events.	Flame Sensor
Motion Sensors	Used to sense the motion and gestures in sensor area environment, this sensor used to monitor the change in the gesture in specific area, the data representation in form of numeric.	PIR Motion sensor

These sensors are grown in the environment by attaching them to the Raspberry pi conductors as shown in the figure (4.1). These sensors monitor and collect data from the environment in which you are located and store it in the Raspberry pi. The sensors make be wired connections using the following circuits (as shown in Figure (4.2)):

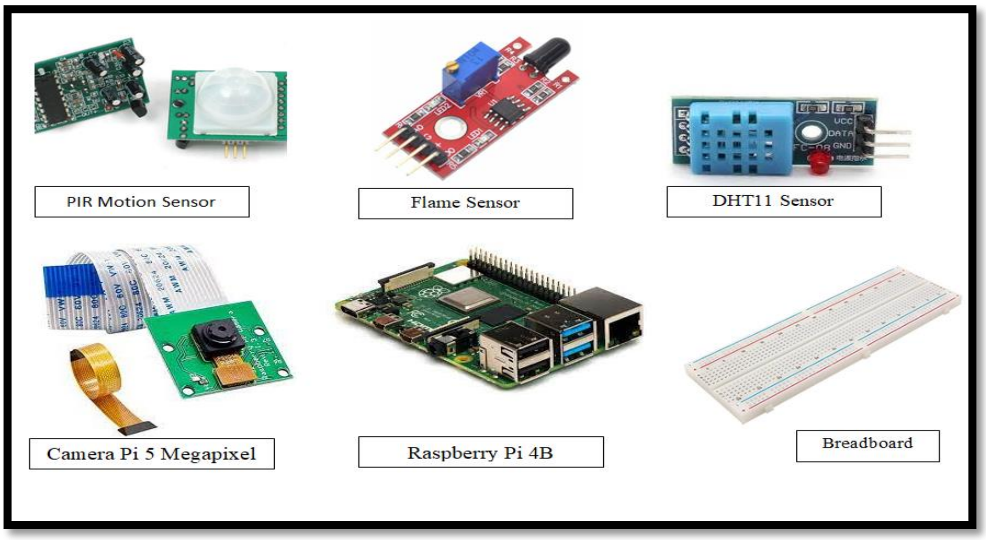


Figure (4.1) Raspberry pi device with sensors attached to it.

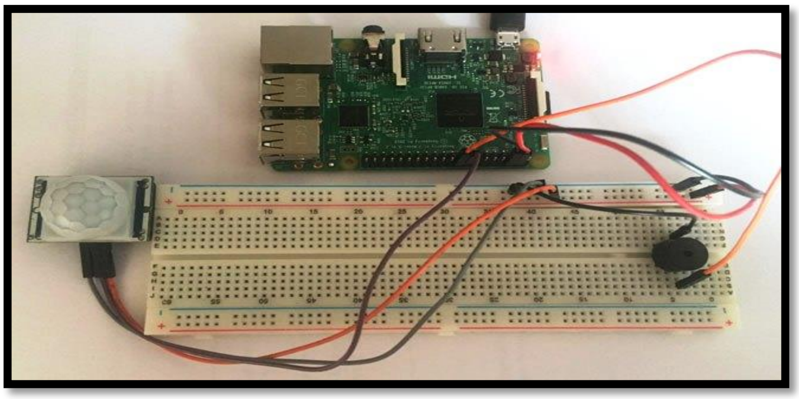


Figure (4.2) Circuit to make sensor work wired.

Raspberry pi does the data collection process from the sensors and then the data is encrypted by the PLVSSM algorithm. The Raspberry pi camera sensor provides an RGB image which is also analyzed and encoded by the PLVSSM algorithm. The encrypted data values are stored in a file for later use. The output of the encrypted data is hashed using modified SHA3 to authenticate the data and ensure that it has not been tampered with and generate the final hash code for the system. The final hash size is fixed at 512 bits and a numeric data model is also mapped to a fixed-size token on the receiver side for authentication verification. The data is obtained synchronously by the Raspberry pi through the wired connection and the Raspberry pi in turn communicates the data continuously. In each chip, there is a time-varying data flow between the sender and receiver that is processed in real-time. The table shows the specifications of the used Raspberry Pi Model 4B.

Table (4.2) The specification of the Raspberry Pi 4B.

Raspberry Pi 4 Model B specification
• Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz • 4GB LPDDR4-3200 SDRAM (depending on model) • 2.4 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE • Gigabit Ethernet • 2 USB 2.0 ports. • Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards) • 2 × micro-HDMI ports (up to 4kp60 supported) • 2-lane MIPI DSI display port • 2-lane MIPI CSI camera port • 4-pole stereo audio and composite video port • H264 (1080p60 decode, 1080p30 encode) • OpenGL ES 3.1, Vulkan 1.0 • Micro-SD card slot for loading operating system and data storage • 5V DC via USB-C connector (minimum 3A*) • 5V DC via GPIO header (minimum 3A*) • Power over Ethernet (PoE) enabled (requires separate PoE HAT) • Operating temperature: 0 – 50 degrees C ambient • A good quality 2.5A power supply can be used if downstream USB peripherals consume less than 500mA in total.

4.4 Data Aggregation Stage

At this point, data is collected from sensors attached to the Raspberry pi that is in the place. The table shows an example of the process of collecting data from sensors. There may be more than one place in which there is also a Raspberry pi device with the sensors attached to it, but this work was limited to a simple model, which is a single Raspberry pi in one place.

Table (4.3) Example of sensing data collection

Date & Time	Temp Sensor (Fahrenheit)	Humidity Sensor (Fahrenheit)	Flame Sensor	PIR Sensor	Camera Sensor (image)	CameraSensor (video)
17:24:54 14/12/2021	21.0	45.0%	Flame Detected	Motion Detected!	Image1	21Dec01_17:24:54.h264
20:17:33 14/12/2021	20.0	47.0%	0	Motion Detected!	Image2	21Dec01_120:17:33.h264
18:12:23 15/12/2021	23.0	62.0%	Flame Detected	0	Image3	21Dec01_18:12:23.h264

As shown in Table (4.3), Raspberry pi collects sensor data during slice time = 1 ms. Where the data indicates on the date 12/14/2021 that the temperature in the room is 22.0 and this is normal, as the humidity is 44%, also there is motion detection and fire. image1 This means that there is an image captured by the camera and stored as image1 as well as a video clip. This sensor reading data will be sent to the PLVSSM algorithms within the Hybrid Authentication and Encryption Algorithms to produce the final cipher - the authentication tokens.

4.5 PLVSSM Implementation and Results

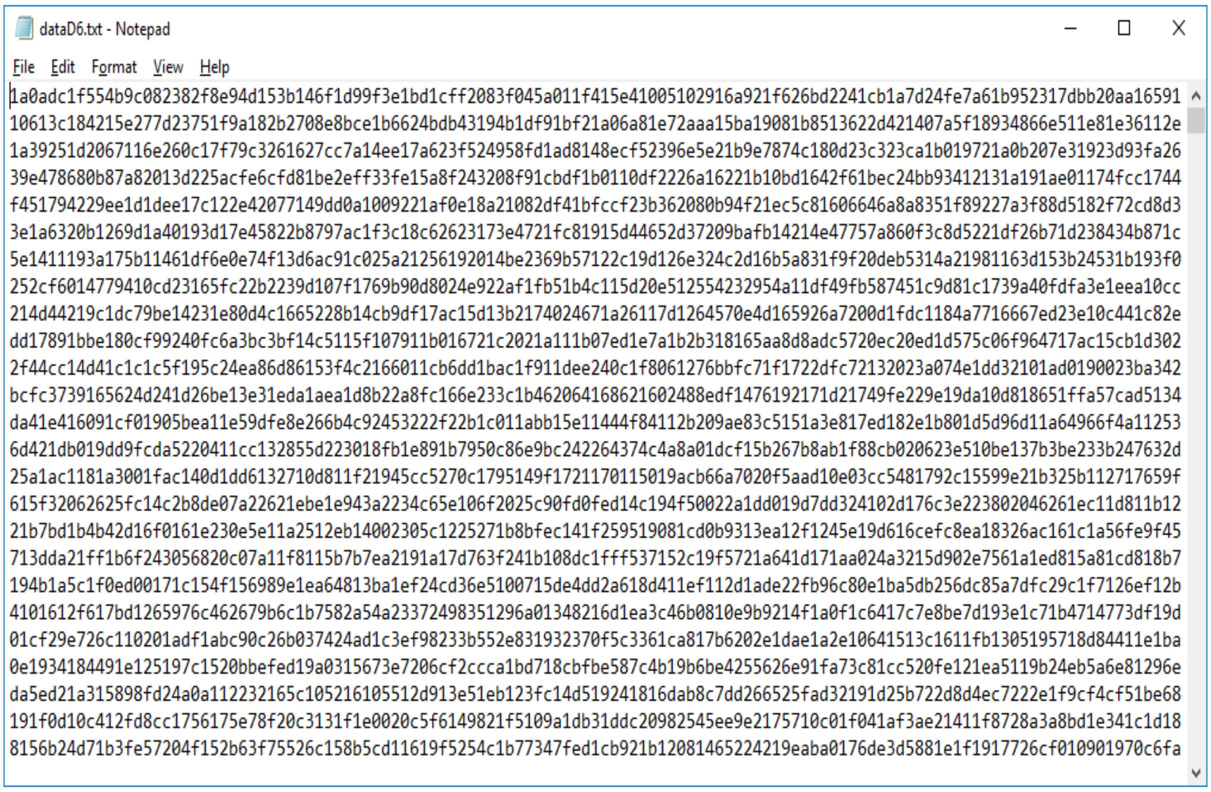
The PLVSSM has three stages: Chaos keys generator, encryption, and authentication. The implementation of each stage was explained bellow:

4.5.1 Chaos Keys Generator

In this work the proposed Chaotic system was used as a generator for Chaos keys and to generate floating numbers for 9-D keys. The first part of the Chaotic system contains a 4-D chaotic system [103] that in turn generates 4-D Chaos keys (K1, K2, K3, and K4). And the second system of 5-D chaotic system [104] that in turn generates 5-D Chaos keys (K5, K6, K7, K 8, and K9). The combination has in the 4D keys were Chaos numbers with 3 positive Lyapunov values and in 5D has 3 positive Lyapunov values. It is clear that the combination made to the chaotic system increased the degree of chaos.

For the second part of the proposed Chaos generator is the Chaotic 5D system [103]. This system is explained in Chapter Three Section 3.4.1. The output of this chaotic system is (K5, K6,… K9) and then this output is then unified with the results of the first part of Chaos generators.

The last 6 digits of each nine key are converted to hexadecimal numbers from the final 9-D Chaotic keys, which are then saved to a file for later use in various stages of encryption and authentication. Figure (4.3) shows an example of 9-D chaos switches used in PLVSSM .



```
1a0adc1f554b9c082382f8e94d153b146f1d99f3e1bd1cff2083f045a011f415e41005102916a921f626bd2241cb1a7d24fe7a61b952317dbb20aa16591
10613c184215e277d23751f9a182b2708e8bce1b6624bdb43194b1df91bf21a06a81e72aaa15ba19081b8513622d421407a5f18934866e511e81e36112e
1a39251d2067116e260c17f79c3261627cc7a14ee17a623f524958fd1ad8148ecf52396e5e21b9e7874c180d23c323ca1b019721a0b207e31923d93fa26
39e478680b87a82013d225acfe6cfd81be2eff33fe15a8f243208f91cbdf1b0110df2226a16221b10bd1642f61bec24bb93412131a191ae01174fcc1744
f451794229ee1d1dee17c122e42077149dd0a1009221af0e18a21082df41bfccf23b362080b94f21ec5c81606646a8a8351f89227a3f88d5182f72cd8d3
3e1a6320b1269d1a40193d17e45822b8797ac1f3c18c62623173e4721fc81915d44652d37209bafb14214e47757a860f3c8d5221df26b71d238434b871c
5e1411193a175b11461df6e0e74f13d6ac91c025a21256192014be2369b57122c19d126e324c2d16b5a831f9f20deb5314a21981163d153b24531b193f0
252cf6014779410cd23165fc22b2239d107f1769b90d8024e922af1fb51b4c115d20e512554232954a11df49fb587451c9d81c1739a40fdfa3e1eea10cc
214d44219c1dc79be14231e80d4c1665228b14cb9df17ac15d13b2174024671a26117d1264570e4d165926a7200d1fdc1184a7716667ed23e10c441c82e
dd17891bbe180cf99240fc6a3bc3bf14c5115f107911b016721c2021a111b07ed1e7a1b2b318165aa8d8adc5720ec20ed1d575c06f964717ac15cb1d302
2f44cc14d41c1c1c5f195c24ea86d86153f4c2166011cb6dd1bac1f911dee240c1f8061276bbfc71f1722dfc72132023a074e1dd32101ad0190023ba342
bcfc3739165624d241d26be13e31eda1aea1d8b22a8fc166e233c1b462064168621602488edf1476192171d21749fe229e19da10d818651ffa57cad5134
da41e416091cf01905bea11e59dfe8e266b4c92453222f22b1c011abb15e11444f84112b209ae83c5151a3e817ed182e1b801d5d96d11a64966f4a11253
6d421db019dd9fcda5220411cc132855d223018fb1e891b7950c86e9bc242264374c4a8a01dcf15b267b8ab1f88cb020623e510be137b3be233b247632d
25a1ac1181a3001fac140d1dd6132710d811f21945cc5270c1795149f1721170115019acb66a7020f5aad10e03cc5481792c15599e21b325b112717659f
615f32062625fc14c2b8de07a22621ebe1e943a2234c65e106f2025c90fd0fed14c194f50022a1dd019d7dd324102d176c3e22380204626 1ec11d811b12
21b7bd1b4b42d16f0161e230e5e11a2512eb14002305c1225271b8bfec141f259519081cd0b9313ea12f1245e19d616cefc8ea18326ac161c1a56fe9f45
713dda21ff1b6f243056820c07a11f8115b7b7ea2191a17d763f241b108dc1fff537152c19f5721a641d171aa024a3215d902e7561a1ed815a81cd818b7
194b1a5c1f0ed00171c154f156989e1ea64813ba1ef24cd36e5100715de4dd2a618d411ef112d1ade22fb96c80e1ba5db256dc85a7dfc29c1f7126ef12b
4101612f617bd1265976c462679b6c1b7582a54a23372498351296a01348216d1ea3c46b0810e9b9214f1a0f1c6417c7e8be7d193e1c71b4714773df19d
01cf29e726c110201adf1abc90c26b037424ad1c3ef98233b552e831932370f5c3361ca817b6202e1dae1a2e10641513c1611fb1305195718d84411e1ba
0e1934184491e125197c1520bbefed19a0315673e7206cf2ccca1bd718cbfbe587c4b19b6be4255626e91fa73c81cc520fe121ea5119b24eb5a6e81296e
da5ed21a315898fd24a0a112232165c105216105512d913e51eb123fc14d519241816dab8c7dd266525fad32191d25b722d8d4ec7222e1f9cf4cf51be68
191f0d10c412fd8cc1756175e78f20c3131f1e0020c5f6149821f5109a1db31ddc20982545ee9e2175710c01f041af3ae21411f8728a3a8bd1e341c1d18
8156b24d71b3fe57204f152b63f75526c158b5cd11619f5254c1b77347fed1cb921b12081465224219eaba0176de3d5881e1f1917726cf010901970c6fa
```

Figure (4.3) An example of the generated 9-D Chaos keys used in PLVSSM.

A- Key Space Analysis

A key space, or keyspace, is the set of all valid, possible, distinct keys of a given cryptosystem. This test is used for preventing the attack called brute force attack. The security of a cryptosystem is proportional to the size of the key space. An intercepted message with a larger keyspace is more resistant to attackers' decryption efforts (cryptanalytic attack). Since an attacker will try to brute force the message with all possible key combinations. An encryption system's key space can range from a small number of combinations to millions. The larger the size of all possible permutations, the stronger the encryption system. In the case of a strong encryption scheme, many keys must be tried in any brute-force attack on that technique. The key or seed-generating chaos is defined by the initial parameters and conditions, from which the key space is obtained. Shannon [105] illustrates, in one of the classic security studies, that the bits needed for the encryption algorithm to be considered as viable for cryptographic applications must be greater than 127 bits. In this work, the key space is found from the initial parameters of two chaotic maps [103][104], which are (x0, y0, z0, w0, a, b, c, d). Thses are double-precision real numbers and they can have 14 decimal digits, so each double-precision real number parameter has 10^{14} possible values. The key space is calculated as $(10^{14})^{8} \approx 2^{372}$, indicating that the keyspace is large and the encryption scheme can withstand brute force attacks. So, this system can be relied upon to produce keys that are more resistant to the attacks mentioned above and then can be relied upon for entering into encryption and decryption operations for the same reason.

4.5.2 The First Proposed Hybrid Lightweight Encryption Algorithm (Speck Hummingbird)

After the process of collecting the data in the Raspberry pi, it is followed by the stage of encoding this data. The encryption phase is based on the use of a hybrid algorithm consisting of two light algorithms, and parts of the algorithms are combined to form a single efficient algorithm. Sensing data collected as a data file is normalized and passed in blocks (with a size of 128 bits per block) to a lightweight hybrid algorithm. A lightweight hybrid algorithm will be applied to acquire data encrypted with the generated Chaos keys (using the proposed Chaotic system). Table (4.4) shows examples of the results of the first proposed encryption algorithm. The first field includes the original image before encryption, followed by the encryption result of the image. After decryption, it is noted that the image matches the original image before the encryption. It is also noted that the three images used are of different sizes. This indicates that the proposed hybrid encryption algorithm can applied on images of different dimensions.

Table (4.4) The examples result of proposed encryption algorithm

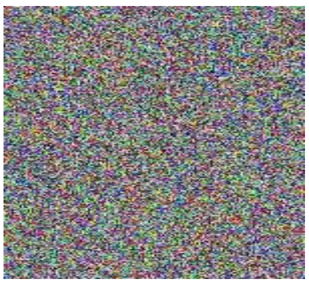
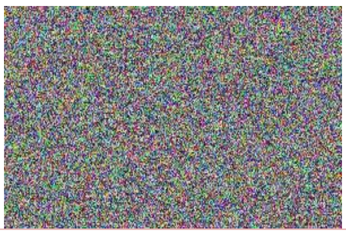
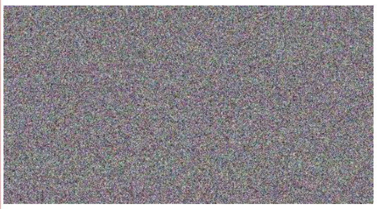
no	original	encrypted	decrypted
S1			
S2			
S3			

Table (4.5) shows the encryption time and decryption time for 7 frames cut from the live broadcast. As these frames are small clips that the camera records when a certain movement occurs. The encoding and decoding time was calculated for each frame from these clips in different sizes 128x128, 256x256, 512x512. It is noted that the larger the dimensions of the frame, the greater the time for encryption and decryption due to the increase in the data of the frame.

Table 4.5: Time consuming for first hybrid encryption / decryption algorithm

Frame size	128x128		256x256		512x512	
operation	Enc. Time (ms)	Dec. Time(ms)	Enc. Time(ms)	Dec. Time(ms)	Enc. Time(ms)	Dec. Time(ms)
1	4.432	4.388	15.988	15.822	43.657	43.630
2	4.643	4.768	15.589	15.399	43.873	43.725
3	4.765	4.654	15.989	15.875	43.865	42.728
4	4.657	4.630	16.198	16.178	43.549	43.335
5	4.547	4.520	16.199	16.107	43.740	43.650
6	4.435	4.403	16.099	16.025	43.465	43.231
7	4.879	4.823	16.109	16.089	43.765	43.698
Average	4.622	4.598	16.024	15.927	43.702	43.428

Table (4.6) shows the encryption time and decryption time also for 7 frames, but in this case, the frame size is 256x256. A different number of rounds is applied to each frame each time (10 rounds 14 rounds 16 rounds). It is known that increasing the number of rounds leads to stronger encryption results, but at the cost of time. It is noticed that at 16 rounds, the encryption and decryption time is higher than at 10 rounds.

***Table 4.6: Time consuming for first hybrid encryption / decryption algorithm with frame size 256x256 pixel and different rounds*.**

Frame size	10 rounds		14rounds		16 rounds	
operation	Enc. Time (ms)	Dec. time(ms)	Enc. time(ms)	Dec. time(ms)	Enc. time(ms)	Dec. time(ms)
1	11.989	11.998	15.099	15.123	17.322	17.257
2	11.895	11.780	15.102	15.543	17.654	17.120
3	11.876	11.799	15.132	15.876	17.965	17.439
4	11.756	11.706	16.364	16.678	17.883	17.947

5	11.325	11.256	16.599	16.546	17.434	17.346
6	11.543	11.433	16.765	16.425	17.049	17.273
7	11.897	11.676	16.798	16.657	17.567	17.254
Average	11.0401	11.092	15.865	16.121	17.839	17.662

In Table (4.7) the evaluation of the first proposed encryption algorithm by NIST standards, where 12 tests were selected out of 15. A test is considered as passed if the p-value is above the significance level of 0.01 or below 0.99. The comparison was made with the hummingbird algorithm as the basic algorithm in the proposal. The comparison was made with different number of rounds for the proposed algorithm (10 rounds 14rounds 16 rounds). The results show that the proposed algorithm has passed all NIST statistical tests. On test "2", "7", "9" and "12" the original algorithm shows a failure to pass the tests, while the proposed algorithm passed the tests with good results.

Table 4.7: NIST Tests of the first hybrid Algorithm

#	Test	Hummingbird algorithm	First Proposed Algorithm (10 round)	First Proposed Algorithm (14 round)	First Proposed Algorithm (16 round)
1	Run Test	0.237676	0.89435	0.93211	0.94563
2	Serial Test	0.006733	0.273912	0.282368	0.29457
3	random excursion variant test	0.798635	0.943125	0.966232	0.975647
4	random excursion test	0.874637	0.98876	0.989883	0.99989
5	non overlapping template matching test	0.990465	0.527673	0.763241	0.778951

6	Frequency Monobit Test	0.2384	0.254104	0.364951	0.476539
7	Maurer’s universal statistical test	0.03477	0.620554	0.743250	0.809564
8	the longest run of ones in a block test	0.34880	0.98826	0.998752	0.99975
9	Linear complexity Test	0.003948	0.956796	0.984210	0.989996
10	Frequency test within a Block test	0.23894	0.883304	0.898975	0.900404
11	Discrete Fourier Transform test	0.639874	0.865436	0.925436	0.935486
12	Cumulative sums Test	0.003943	0.235467	0.398975	0.405379

4.5.3 The second hybrid encryption

The second hybrid algorithm was applied to the same data collected by Raspberry Pi. The results showed that the second suggestion of the algorithm gave results in encryption and decryption slower than the first suggestion, but the randomness increased because the main algorithm is the speck algorithm. Table (4.8) shows the examples result of proposed encryption algorithm.

Table (4.8) The examples result of proposed encryption algorithm

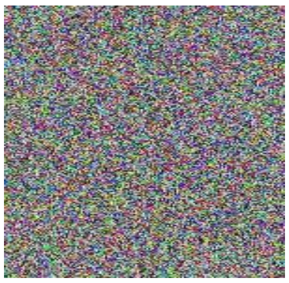
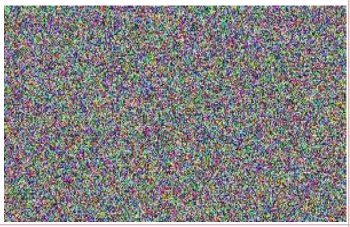
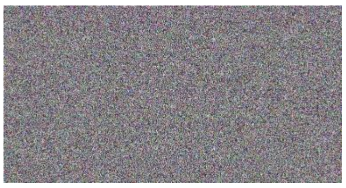
no	original	encrypted	decrypted
S1			
S2			
S3			

Table (4.9) illustrates the encryption and decryption times for 7 frames with different sizes. Compared with the results of the first proposal for the same image size at 128x128, for example, in the first proposal, the encryption speed was 4.432. While in the second proposal for the same data size, the encryption speed is 16.449. There is a clear and big difference in speed because the basic algorithm is the speck algorithm. As it is known, the algorithm structure is block cipher, which is much slower than the stream cipher.

Table 4.9: Time consuming for second hybrid encryption / decryption algorithm

frame size	128x128		256x256		512x512	
operation	Enc. Time (ms)	Dec. Time(ms)	Enc. Time(ms)	Dec. Time(ms)	Enc. Time(ms)	Dec. Time(ms)
1	16.449	15.672	33.321	30.232	79.930	74.654
2	16.142	15.290	33.301	30.582	82.897	76.980
3	16.320	15.824	33.452	30.632	82.750	76.760
4	16.488	14.423	33.634	30.502	82.655	76.087
5	17.332	14.265	33.752	30.654	82.532	75.123
6	17.543	15.025	33.540	30.757	82.121	75.225
7	16.842	15.150	33.722	31.682	82.098	75.324
Average	16.730857	15.092714	33.81743	30.72014	82.1404	75.7361

Table (4.10) also shows the encryption and decryption average of the combined 7 frames. The results of encryption and decryption were calculated a different number of rounds each time, as shown in the table (4.10). The results of the proposed algorithm were compared with the results of the original speck algorithm. The encryption time of the proposed algorithm is calculated with a different number of rounds each time (10 round, 14 round , 16 round).

Table 4.10: Time consuming for second hybrid encryption / decryption algorithm with frame size 256x256 pixel and different rounds.

frame size	10 rounds		14rounds		16 rounds	
operation	Enc. Time (ms)	Dec. time(ms)	Enc. time(ms)	Dec. time(ms)	Enc. time(ms)	Dec. time(ms)
1	29.490	28.902	34.009	31.079	38.224	37.321
2	29.220	28.225	33.198	31.087	38.345	37.520
3	29.620	28.987	34.098	31.076	38.458	37.009
4	29.489	28.254	34.760	31.387	38.542	37.019
5	29.023	28.420	34.599	31.370	38.675	37.120
6	29.564	28.321	34.529	30.870	38.722	37.980
7	29.854	28.189	34.427	31.011	38.876	37.064
Average	29.751	28.756	34.231	31.125	38.406	37.147

This proposal also passed all NIST statistical tests as shown in Table (4.11). The original algorithm (speck) failed the tests "2", "9" and "10". While the proposed algorithm has passed these tests.

Table 4.11: NIST Tests of the Second hybrid Algorithm

#	Test	Speck algorithm	second Proposed Algorithm(10 round)	second Proposed Algorithm(14 round)	Second Proposed Algorithm(16 round)
1	Run Test	0.4443	0.983241	0.99811	0.99875
2	Serial Test	0.00755	0.362801	0.382128	0.411873
3	random excursion variant test	0.89760	0.954341	0.967431	0.977324
4	random excursion test	0.45223	0.989981	0.998992	0.999891

5	non overlapping template matching test	0.88789	0.648732	0.786541	0.793495
6	Frequency Monobit Test	0.18870	0.354302	0.375341	0.486529
7	Maurer's universal statistical test	0.44324	0.540764	0.643653	0.708594
8	The longest run of ones in a block test	0.39802	0.998763	0.998672	0.99985
9	Linear complexity Test	0.000786	0.987893	0.988760	0.989967
10	Frequency test within a Block test	0.00887	0.894304	0.899495	0.900567
11	Discrete Fourier Transform test	0.79865	0.896542	0.926736	0.946756
12	Cumulative sums Test	0.01332	0.343567	0.418975	0.435369

4.5.4 The Third proposed hybrid encryption

On the same data stored in Raspberry Pi on which the first and second suggested methods were applied, the third proposed algorithm is applied. It is as shown in the results that it is faster than the previous two methods because the two algorithms work in parallel at the same time. This mechanism of action shortens the time a lot and gives very fast results. Table (4.12) shows the examples result of proposed encryption algorithm.

Table (4.12) The examples result of proposed encryption algorithm

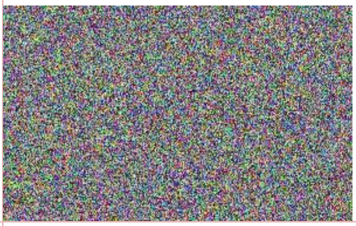
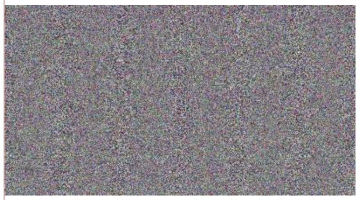
no	original	encrypted	decrypted
S1			
S2			
S3			

Table (4.13) illustrates the speed of encryption and decryption for the third proposal with different image sizes for 7 frames. Each frame has a different encryption and decryption speed because the pixel information in the frame varies from one to another according to the movement in the frame. At the end of the table, there is an average speed of 7 frames combined.

Table 4.13: Time consuming for third hybrid encryption / decryption algorithm

Image size	128x128		256x256		512x512	
operation	Enc. Time (ms)	Dec. time(ms)	Enc. time(ms)	Dec. time(ms)	Enc. time(ms)	Dec. time(ms)
1	3.321	3.277	13.876	12.876	31.437	30.535
2	3.532	3.657	13.475	12.546	31.523	30.626
3	3.654	3.543	13.876	12.534	31.725	30.627
4	3.546	3.520	14.256	14.456	31.669	30.236
5	3.436	3.410	14.240	14.327	31.570	30.453
6	3.324	3.332	14.120	14.219	31.485	30.135
7	3.768	3.712	14.132	14.128	31.325	30.644
Average	3.511	3.493	13.996	13.583	31.533	30.465

Table (4.14) shows the encryption and decryption speed of 256 * 256 frame size with a different number of rounds each time, the number of rounds is reduced to make the algorithm run faster.

Table 4.14: Time consuming for third hybrid encryption / decryption algorithm with frame size 256x256 pixel and different rounds

frame size	10 rounds		14 rounds		16 rounds	
operation	Enc. Time (ms)	Dec. time(ms)	Enc. time(ms)	Dec. time(ms)	Enc. time(ms)	Dec. time(ms)
1	11.989	11.822	13.180	13.554	16.123	15.557
2	11.895	11.750	13.145	13.419	15.454	15.720
3	11.876	11.702	13.135	13.089	16.865	15.739
4	11.756	11.698	13.278	13.178	16.783	16.985
5	11.325	11.203	13.452	13.354	16.334	16.566
6	11.543	11.398	13.564	13.433	16.039	16.543
7	11.897	11.760	13.432	13.329	16.437	15.764
Average	11.183	11.0475	13.312	13.336	16.719	15.982

As in the first and second proposals, the third proposed algorithm passed all NIST statistical tests as shown in the table (4.15).

4.15: NIST Tests of the third hybrid Algorithm

#	Test	Speck algorithm	Hummingbird algorithm	Third Proposed Algorithm(10 round)	Third Proposed Algorithm (14 round)	Third Proposed Algorithm (16 round)
1	Run Test	0.237676	0.237676	0.872833	0.884313	0.93321
2	Serial Test	0.006733	0.006733	0.178872	0.179792	0.246758
3	random excursion variant test	0.798635	0.798635	0.934627	0.87927	0.897325
4	random excursion test	0.874637	0.874637	0.88785	0.914235	0.964328
5	non overlapping template matching test	0.990465	0.990465	0.40871	0.426871	0.546731
6	Frequency Monobit Test	0.002384	0.2384	0.175843	0.187104	0.247367
7	Maurer's universal statistical test	0.000034	0.03477	0.574784	0.587354	0.643123
8	the longest run of ones in a block test	0.000348	0.34880	0.87626	0.93286	0.987659
9	Linear complexity Test	0.039485	0.003948	0.84396	0.872396	0.893510
10	Frequency test within a Block test	0.023894	0.23894	0.734504	0.765504	0.796065

11	Discrete Fourier Transform test	0.639874	0.639874	0.652236	0.698736	0.760531
12	Cumulative sums Test	0.003943	0.003943	0.134277	0.247657	0.387907

4.6 Hamming distance Analysis

Hamming distance is used to measure the similarity or dissimilarity between two sequences. Used in Software and Hardware decision decoding. Hamming distance between two words a=(a_0,a_1, … ,a_{n-1}) and b=(b_0,b_1,…,b_{n-1}) in Galois Field GF(2), is the number of coordinates in which the two blocks differ.

$$dHamming=dH(a,b)=\#\{j:aj\neq bj,j=0,1,\cdots,n-1\}.$$

For example, the Hamming distance between (0,0,0,1) and (1,0,1,0) in GF(2) is 3, since they differ in three digits.

The hamming distance of the test image is that encrypted by two keys using a proposed algorithm should be the difference in total bits when finding Hamming distance as shown in table (4.16). The results prove that the hamming distance of the two proposed is secure and able to resist statistical attacks.

Table 4.16: Hamming distance Results of Encrypted images

Text Size (byte)	SPECK	Hummingbird	First hybrid algorithm	Second hybrid algorithm	Third hybrid hybrid algorithm
128*128	11521	10663	17563	19462	20752
256*256	51790	50998	76231	78409	80509
512*512	162109	154434	293289	302531	324598

4.7 Image Quality Test

There are several objective tests for qualifying the similarity of two images and this test may be used for finding the dissimilarity between them. The image Quality was tested for encrypted images concerning the original image by finding the difference to assure image quality. Mean square error (MSE) denoted the high value between the original and encrypted images. Another test depends on MSE used such as signal-to-noise ratio (SNR) and peak signal-to-noise ratio (PSNR) these two tests are denoted by minimizing value and the last test is structural similarity index (SSIM) used to find the interior correlation on image objects. Also, the entropy test is applied to the encrypted image and the value of all images is near the maximum required bits (8 bits) for all images. Tables 4.17-4.19 explains these tests in detail for image set 1 for the three proposed hybrid algorithms.

Table 4.17: Image quality test for set 1 using first hybrid encryption algorithm.

#	MSE	PSNR	SNR	SIM	Entropy
1	9871.44698	0.00401	2.32483	113.82103	7.98723
2	8659.68729	0.00434	1.89688	110.75863	7.98748
3	9853.57029	0.00452	1.54778	143.78329	7.98764
4	9906.69612	0.00466	2.10949	112.76748	7.98772
5	9034.32273	0.00498	1.97565	145.54257	7.98735
6	9869.8341	0.00482	1.80787	132.18569	7.98798
7	8949.70713	0.00501	2.20594	133.64545	7.98883

Table 4.18: Image quality test for set 1 using second hybrid encryption algorithm.

#	MSE	PSNR	SNR	SIM	Entropy
1	9793.45898	0.00419	1.98321	102.86596	7.99830
2	8899.78640	0.00456	1.50459	98.17480	7.99850
3	9920.77123	0.00363	1.34198	120.59209	7.99863
4	10234.84112	0.00349	1.89054	108.45485	7.99872
5	9644.45325	0.00439	1.86565	127.92110	7.99865
6	9890.54539	0.00418	1.74598	101.9328	7.99845
7	9560.56726	0.00429	1.32474	113.78267	7.99765

Table 4.19: Image quality test for set 1 using third hybrid encryption algorithm.

#	MSE	PSNR	SNR	SIM	Entropy
1	9923.43498	0.00420	1.87331	109.87796	7.98967
2	9189.76740	0.00463	1.46019	104.11980	7.99409
3	10890.75323	0.00370	1.231098	137.57109	7.99671
4	10450.85812	0.00350	1.87754	118.40785	7.99811
5	9674.47625	0.00440	1.78965	131.93310	7.99870
6	9987.55239	0.00420	1.68098	114.9009	7.99873
7	9378.57426	0.00430	1.198774	124.79087	7.99755

From the previous tables, the value of the tested image explained the high differences between the original and encrypted image. It denoted that no correlation between input and output image.

4.8 The Proposed Authentication in PLVSSM

The authentication proposed in PLVSSM for IoT contains the proposed lightweight algorithm cipher stage and the modified SHA3-256 stage. After collecting the data from the sensors, we apply the proposed hashing lightweight (512-bit) segmentation for all sensors.

4.8.1 Modified SHA3-256

In the third chapter, in the algorithm (3.7) the steps of Hybrid SHA3-256 are explained. Modifications were made to SHA3 to get more randomness due to the transformation process. The functionality of SHA3-256 is compared with SFHASH3-256 as shown in the table (4.20).

For example, when the data is "000000000", the hash value when used SHA3-256 is "7E1890AE2F6097F0BC8D2CA59". But the result will be different for the same data using the proposed hybrid authentication system. This gives strength to the results as they are unpredictable.

Table (4.20) SFHASH3 and SHA3 Results Comparison

DATA	SFHASH3-256	SHA3-256
000000000000 00000	A018F4F11C1E836D990BD2 E6FC6A3F4E56B04703D389 4DA0001F5C9B14C8D3B6	7E1890AE2F6097F0BC8D2CA59 75A8819D74CE9EABAD6AE1B5 B4446EAE04A3ABB
0123456789A BCDEF	8AA03AFBEFE965CA18662 1D59DE194538EC95DC57C A8AC00544AFB5407304CC 8	A5DF4CAAE9FDB5DBACF6670 75B709A2F30A115C43168AF332 062B42D4B0DA01F

FFFFFFFF FF FFFFFF	218C8E1FB78138F2817F0B8 E7837C2C87025BF3727A63 C842DF12D2A8919F648	2A8BBC71BCF6442EC96DB96F4 77345AA38050FFE7B88F15A35E 991912C6DC81A
111111111 111 1111	97CF34D1FE61A87A1C5C0 741159172CA8129E3D0641 CF9F8DE215D673790A484	1AFD827639BD0919C0F788EB1 C9F80AAABD13AC91949610CB DBBCA909401DD14
101010101 010 1010	029CA03AE25A1A28E676F BE78E0C48794B568C267F5 B14BF5EEABD85983BEA53	002E4343C2E457947C1E3F9F188 878F9C44C1C0B52FE7FC71DDA B0A066E010EB
000000001 111 1111	F791EB276F260946954E5FB 3F93590F4B744C634A6F0D AB3B34270916E7FE14A	EDFCA6BFA232070876F9E7F49 E431CBDB02E3C7981CE5801470 B56503DF36AA0

Table (4.21) shows the comparison hashing time. As shown, the SFHASH3-256 has little more time in its operation.

Table (4.21) Comparison of Hashing Time for SFHASH3

Text Size (Byte)	SFHASH3-256 (msec)	Throughput (byte/msec)	SHA3-256 (msec)	Throughput (byte/msec)
10	0.0067	1492.54	0.0069	1449.2754
25	0.0098	2551.02	0.0101	2475.2475
70	0.0110	6363.64	0.0121	5785.124
100	0.0123	8130.08	0.0131	7633.5878
1000	0.0467	21413.3	0.0512	19531.25

2000	0.0788	25380.7	0.0843	23724.792
10000	0.1110	90090.1	0.1233	81103.001
500000	9.2131	54270.5	9.3034	53743.793
1000000	12.1210	82501.4	13.9705	71579.399

4.9 Results Discussion

Relying on the results obtained from testing the system and the evaluations presented in the previous sections, the objectives of this thesis have been achieved and satisfied. The goals of the system have been achieved, which are to provide protection for the place and respond to emergency events through the proposed algorithms for the system.

1. The proposed hybrid algorithm that resulted from merging parts of the Hummingbird and Speck algorithm increases the strength of the security encryption with an acceptable fast encryption/decryption time. It also makes the algorithm suitable for embedding inside the sensors to achieve power consumption.
2. As a result of the tests, it was confirmed that the proposed hybrid algorithm passed the NIST statistical tests for encryption of different volumes of data.
3. The use of SHA3-256 authentication in the hybrid algorithm increased the randomness of the encrypted result of the data.
4. The modifications to Chaotic system have to change the Lyapunov extensions to be positive values in the specific parameter parameters, make results key randomization and powerful to use in the encryption algorithm, and be more satisfying to embedded into IoT devices and sensors for power-consuming due to fasting keys generation.

5. The proposed Chaotic system has a positive Lyapunov extension for 4- D, causing more security to all encryption/ authentication algorithms used.
6. Using PLVSSM operation help in avoiding many attacks and reduce the false alarm ratio in emergency response by keeping transfer the correct (actual) data, and avoiding any manipulation.

Chapter Five

Conclusions and Suggestions for Future Works

Chapter Five
Conclusions and Suggestions for Future Works

5.1 Conclusions

From the tests results and the evaluations of the proposed system which is presented in the previous chapter, the goals of this dissertation were achieved and satisfied. The confidentiality, integrity and encryption of video data from unauthorized access. Were attained and accomplished by the proposed algorithms in the proposed system. After the proposed system has been implemented, there are some conclusions that are clarified as follows:

1. The first suggestion of the hybrid algorithm gave rather fast results, while the second suggestion had slower results, but it was characterized by more randomness. The third proposal was faster than the previous two proposals and gave good results in encryption and decryption.
2. The proposed algorithms passed all NIST statistical tests for encryption different data volumes. That's why the algorithms can avoid a brute force attack.
3. The system must be fast in sending data to the responsible party. As the slightest delay may cause many problems, especially that the video clips do not bear a delay. The delay may lead to distortion of the clip and the loss of important data. For this reason, algorithms that are faster than traditional algorithms were chosen.
4. The system has increased data security by authenticating the data and verifying the identity of the user using the developed Hash system (SHA3).
5. Strong and random keys were obtained for use in the encryption algorithms. This was due to the use of positive Lyapunov extensions of the 9D combination chaotic system.

6. The chaos keys generated by the proposed 9D chaos system are more suitable for inclusion within IoT devices and sensors. It also provided more security when using the chaotic system outputs proposed on the encryption/authentication algorithms.

5.2 Future Works

There is a set of suggestions for future work by which the proposed system can be tested, evaluated and improved. Which can be listed as follows:

1. Groups of sensors can be placed in different places, all connected to the Raspberry Pi and a camera. All sensors information is collected in the cloud.
2. An intelligent Encryption algorithm can be used in data encryption layer in order to increase the security issue.
3. An intelligent Hashing technique can be embedding to the authentication layer to decreased the hashing time.
4. The fog and cloud computing can used to support the PLVSSM different levels for storing data, keys, and parameters.

References

References

[1] Q. I. Sarhan, **"Internet of things: a survey of challenges and issues"**, *Int. J. Internet Things Cyber-Assurance,* vol. 1, no. 1, p. 40, 2018, doi: 10.1504/ijitca.2018.10011246.

[2] M. H. Ali and N. K. Ali, **"IoT based security system and intelligent home automation multi monitoring and control systems",** *IAES Int. J. Robot. Autom.*, vol. 8, no. 3, p. 205, 2019, doi: 10.11591/ijra.v8i3.pp205-210.

[3] R. Pereira and E. Pereira, "**Video streaming: H.264 and the internet of things**," *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, 2015.

[4] C. W. Chen, **"Internet of Video Things: Next-Generation IoT With Visual Sensors,"** in IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6676-6685, Aug. 2020, doi: 10.1109/JIOT.2020.3005727.

[5] R. S. Mohammed, A. H. Mohammed, and F. N. Abbas, **"Security and Privacy in the Internet of Things (IoT): Survey",** *2nd Int. Conf. Electr. Commun. Comput. Power Control Eng. ICECCPCE 2019*, no. February 2021, pp. 204–208, 2019, doi: 10.1109/ICECCPCE46549.2019.203774.

[6] S. B. Sadkhan and A. O. Salman, "**A survey on lightweight-cryptography status and future challenges,"** 2018 International Conference on Advance of Sustainable Engineering and its Application (ICASEA), 2018, pp. 105-108, doi: 10.1109/ICASEA.2018.8370965.

[7] S. B. Sadkhan and Z. Hamza, **"Cryptosystems used in IoT-current status and challenges,"** 2017 International Conference on Current Research in Computer Science and Information Technology (ICCIT), 2017, pp. 58-62, doi: 10.1109/CRCSIT.2017.7965534.

[8] B. F. N. M. Alabassby, J. F. Mahdi, and M. A. Kadhim, **"Design and implementation WSN based on Raspberry Pi for medical application",** in *IOP Conference Series: Materials Science and Engineering*, 2019, vol. 518, no. 5, p. 52022.

[9] V. Bhuvaneswari, R Porkodi “***The Internet of Things (IoT)Applications and Communication Enabling Technology Standards: An Overview***” International Conference on Intelligent Computing Applications, IEEE DOI 10.1109/ICICA.2014.73, 2014

[10] A. A. . Abdallah and A. K. . Farhan, “**A New Image Encryption Algorithm Based on Multi Chaotic System”,** *eijs*, vol. 63, no. 1, pp. 324–337, Jan. 2022.

[11] L. A. Hamood and M. K. Ibrahem, **"Video Encryption Based on Chaotic System and Stream Cipher",** *Iraqi J. Inf. Commun. Technol.*, vol. 1, no. 2, pp. 33–40, 2018, doi: 10.31987/ijict.1.2.19.

[12] K. Muhammad, R. Hamza, J. Ahmad, J. Lloret, H. Wang, and S. W. Baik, “***Secure Surveillance Framework for IOT systems using Probabilistic Image Encryption***,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3679–3689, 2018.

[13] Z. M. Jawad Kubba and H. K. Hoomod, “**A hybrid modified lightweight algorithm combined of two cryptography algorithms present and Salsa20 using chaotic system**,” *2019 First International Conference of Computer and Applied Sciences (CAS)*, 2019.

[14] J. R. Naif, G. H. Abdul-Majeed, and A. K. Farhan, **“Secure IOT system based on Chaos-Modified Lightweight AES,”** *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 2019.

[15] A. A. Abd Ulkadhim, “Chaos-modified lightweight present algorithm for Image Encryption,” *International Journal of Engineering Research and Advanced Technology*, vol. 06, no. 02, pp. 26–30, 2020.

[16] M. A. Al-Husainy, “Secure and lightweight encryption model for IOT Surveillance Camera,” *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 2, pp. 1840–1847, 2020.

[17] F. Masood *et al.*, ‘A lightweight chaos-based medical image encryption

scheme using random shuffling and XOR operations’, *Wirel. Pers. Commun.*, pp. 1–28, 2021.

[18] R. Denis and P. Madhubala, ‘Hybrid data encryption model integrating multi-objective adaptive genetic algorithm for secure medical data communication over cloud-based healthcare systems’, *Multimed. Tools Appl.*, vol. 80, no. 14, pp. 21165–21202, 2021.

[19] H. Mohammad and A. Abdullah, ‘Enhancement process of AES: a lightweight cryptography algorithm-AES for constrained devices’, *TELKOMNIKA (Telecommunication Comput. Electron. Control.*, vol. 20, p. 551, Jun. 2022, doi: 10.12928/telkomnika.v20i3.23297.

[20] V. I. S. H. A. L. A. THAKOR, M. A. RAZZAQUE, and M. U. H. A. M. M. A. D. R. A. KHANDAKER, “(PDF) Lightweight Cryptography for IOT: A state-of-the-art,” *Lightweight Cryptography for IoT: A State-of-the-Art*. [Online]. Available: https://www.researchgate.net/publication/342434954_Lightweight_Cryptography_for_IoT_A_State-of-the-Art. [Accessed: 20-Mar-2022].

[21] V. A. Thakor, M. A. Razzaque, and M. R. Khandaker, “Lightweight cryptography algorithms for resource-constrained IOT devices: A review, comparison and research opportunities,” *IEEE Access*, vol. 9, pp. 28177–28193, 2021.

[22] N. Kareem Jumaa, “Survey: Internet of thing using FPGA,” *Iraqi Journal for Electrical And Electronic Engineering*, vol. 13, no. 1, pp. 38–45, 2017.

[23] G.Jayavardhana, B.Rajkumar, S. Marusic, and M. Palaniswami, "Internet of Things: A Vision, Architectural Elements, and Future Directions. Future Generation", (2013).

[24] X. Zhai, X. Guan, C. Zhu, L. Shu and J. Yuan, "Optimization algorithms for multiaccess green communications in Internet of Things", *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1739-1748, Jun. 2018.

[25] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, “Internet of things (IOT) security: Current status, challenges and prospective measures,” *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2015.

[26] B. Alhayani, H. Jasim Mohammed, I. Zeghaiton Chaloob, and J. Saleh Ahmed, ‘Effectiveness of artificial intelligence techniques against cyber security risks apply of IT industry’, *Mater. Today Proc.*, 2021, doi: 10.1016/j.matpr.2021.02.531.

[27] K. O. M. Salih, T. A. Rashid, D. Radovanovic, and N. Bacanin, ‘A Comprehensive Survey on the Internet of Things with the Industrial Marketplace’, *Sensors*, vol. 22, no. 3, 2022, doi: 10.3390/s22030730.

[28] S. B. Sadkhan and Z. Salam, ‘Security and Privacy in Internet of Things-Status, Challenges’, in *2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA)*, 2021, pp. 308–312.

[29] V. Aleksandrovičs, E. Filičevs, and J. Kampars, ‘Internet of things: Structure, features and management’, *Inf. Technol. Manag. Sci.*, vol. 19, no. 1, pp. 78–84, 2016.

[30] T. Yashiro, S. Kobayashi, N. Koshizuka, and K. Sakamura, ‘An Internet of Things (IoT) architecture for embedded appliances’, in *2013 IEEE Region 10 Humanitarian Technology Conference*, 2013, pp. 314–319.

[31] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, ‘IoT middleware: A survey on issues and enabling technologies’, *IEEE Internet Things J.*, vol. 4, no. 1, pp. 1–20, 2016.

[32] F. Muhammad, W. Anjum, and K. S. Mazhar, ‘A critical analysis on the security concerns of internet of things (IoT)’, *Int. J. Comput. Appl.*, vol. 111, no. 7, pp. 1–6, 2015.

[33] M. A. M. Sadeeq, S. R. M. Zeebaree, R. Qashi, S. H. Ahmed, and K. Jacksi, ‘Internet of Things security: a survey’, in *2018 International Conference on Advanced Science and Engineering (ICOASE)*, 2018, pp. 162–166.

[34] R. Kumar, T. W. Au, Wan, W. Susanty, and H. Suhaili, ‘Exploring Data Security and Privacy Issues in Internet of Things Based on Five-Layer Architecture’, *Int. J. Commun. Networks Inf. Secur.*, vol. 12, pp. 108–121, Apr. 2020, doi: 10.17762/ijcnis.v12i1.4345.

[35] A. Shifa, M. N. Asghar, and M. Fleury, ‘Multimedia security perspectives in IoT’, *2016 6th Int. Conf. Innov. Comput. Technol. INTECH 2016*, vol. 7, no. 4, pp. 550–555, 2017, doi: 10.1109/INTECH.2016.7845081.

[36] A. B. F. Khan and G. Anandharaj, ‘A Multi-layer Security approach for DDoS detection in Internet of Things’, *Int. J. Intell. Unmanned Syst.*, 2020.

[37] H. A. Khattak, M. A. Shah, S. Khan, I. Ali, and M. Imran, ‘Perception layer security in Internet of Things’, *Futur. Gener. Comput. Syst.*, vol. 100, pp. 144–164, 2019.

[38] S. Gautam, A. Malik, N. Singh, and S. Kumar, 'Recent advances and countermeasures against various attacks in IoT environment', in *2019 2nd International Conference on Signal Processing and Communication (ICSPC),* 2019, pp. 315–319.

[39] T. Aziz and E. Haq, 'Security Challenges Facing IoT Layers and its Protective Measures', *Int. J. Comput. Appl.*, vol. 179, no. 27, pp. 31–35, 2018, doi: 10.5120/ijca2018916607.

[40] O. Bello, S. Zeadally, and M. Badra, 'Network layer inter-operation of Device-to-Device communication technologies in Internet of Things (IoT)', *Ad Hoc Networks*, vol. 57, pp. 52–62, 2017.

[41] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, 'A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications', *IEEE internet things J.*, vol. 4, no. 5, pp. 1125–1142, 2017.

[42] M. M. Ahemd, M. A. Shah, and A. Wahid, 'IoT security: A layered approach for attacks & defenses', in *2017 international conference on Communication Technologies (ComTech),* 2017, pp. 104–110.

[43] G. Nebbione and M. C. Calzarossa, 'Security of IoT application layer protocols: Challenges and findings', *Futur. Internet*, vol. 12, no. 3, p. 55, 2020.

[44] S. A. Kumar, T. Vealey, and H. Srivastava, 'Security in internet of things: Challenges, solutions and future directions', in *2016 49th Hawaii International Conference on System Sciences (HICSS),* 2016, pp. 5772–5781.

[45] G. Avoine, M. A. Bingöl, X. Carpent, and S. B. O. Yalcin, 'Privacy-friendly authentication in RFID systems: on sublinear protocols based on symmetric-key cryptography', *IEEE Trans. Mob. Comput.*, vol. 12, no. 10, pp. 2037–2049, 2012.

[46] A. E. Barron, 'An overview of digital video', *J. Comput. High. Educ.*, vol. 7, no. 1, pp. 69–84, 1995.

[47] T. Schlogl, C. Beleznai, M. Winter, and H. Bischof, 'Performance evaluation metrics for motion detection and tracking', in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2004, vol. 4, pp. 519–522.

[48] B. Lee and M. Hedley, 'Background estimation for video surveillance', 2002.

[49] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, 'Foreground object detection from videos containing complex background', in *Proceedings of the eleventh ACM international conference on Multimedia,* 2003, pp. 2–10.

[50] C. P. Papageorgiou and T. Poggio, ‘A trainable object detection system: Car detection in static images’, 1999.

[51] Wang, B. Li, Y. Zhang, and J. Yang, “Background modeling and referencing for moving cameras-captured surveillance video coding in HEVC,” *IEEE Trans. Multimed.*, vol. 20, no. 11, pp. 2921–2934, 2018.

[52] S. F. Hassan and R. Fareed, ‘Video streaming processing using fog computing’, in *2018 International Conference on Advanced Science and Engineering (ICOASE)*, 2018, pp. 140–144.

[53] R. Pereira and E. G. Pereira, ‘Video streaming considerations for internet of things’, in *Proceedings - 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014*, Dec. 2014, pp. 48–52. doi: 10.1109/FiCloud.2014.18.

[54] R. Jabbar, M. Shinoy, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, ‘Urban traffic monitoring and modeling system: An iot solution for enhancing road safety’, in *2019 international conference on internet of things, embedded systems and communications (iintec)*, 2019, pp. 13–18.

[55] U. Ashraf, ‘PROSE–Proactive resilience in Internet of Things: targeted attacks and countermeasures’, *IEEE Sens. J.*, vol. 18, no. 24, pp. 10049–10057, 2018.

[56] M. M. Saleh, ‘WSNs and IoT Their Challenges and applications for Healthcare and Agriculture: A Survey.’, *Iraqi J. Electr. Electron. Eng.*, 2020.

[57] J. Wang, C. Lu, M. Wang, P. Li, S. Yan, and X. Hu, ‘Robust face recognition via adaptive sparse representation’, *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2368–2378, 2014.

[58] W. Hamidouche, M. Farajallah, N. Sidaty, S. El Assad, and O. Deforges, ‘Real-time selective video encryption based on the chaos system in scalable HEVC extension’, *Signal Process. Image Commun.*, vol. 58, pp. 73–86, 2017, doi: 10.1016/j.image.2017.06.007.

[59] M. Preishuber, T. Hütter, S. Katzenbeisser, and A. Uhl, ‘Depreciating motivation and empirical security analysis of chaos-based image and video encryption’, *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 9, pp. 2137–2150, 2018.

[60] A. Rego, A. Canovas, J. M. Jimenez, and J. Lloret, ‘An Intelligent System for Video Surveillance in IoT Environments’, *IEEE Access*, vol. 6, pp. 31580–31598, Jun. 2018, doi: 10.1109/ACCESS.2018.2842034.

[61] Y. Wang, J. Xu, and W. Ji, ‘A Feature-based Video Transmission Framework for Visual IoT in Fog Computing Systems’, in *2019 ACM/IEEE*

Symposium on Architectures for Networking and Communications Systems (ANCS), 2019, pp. 1–8.

[62] F. Liu and H. Koenig, ‘A survey of video encryption algorithms’, *Comput. Secur.*, vol. 29, no. 1, pp. 3–15, 2010.

[63] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, ‘Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities’, *IEEE Access*, vol. 9, pp. 28177–28193, 2021, doi: 10.1109/ACCESS.2021.3052867.

[64] S. F. Aghili, H. Mala, P. Kaliyar, and M. Conti, ‘SecLAP: Secure and lightweight RFID authentication protocol for Medical IoT’, *Futur. Gener. Comput. Syst.*, vol. 101, pp. 621–634, 2019.

[65] P. Panahi, C. Bayılmış, U. Çavuşoğlu, and S. Kaçar, ‘Performance Evaluation of Lightweight Encryption Algorithms for IoT-Based Applications’, *Arab. J. Sci. Eng.*, vol. 46, no. 4, pp. 4015–4037, 2021, doi: 10.1007/s13369-021-05358-4.

[66] N. M. Naser and J. R. Naif, ‘A systematic review of ultra-lightweight encryption algorithms’, *Int. J. Nonlinear Anal. Appl.*, vol. 13, no. 1, pp. 3825–3851, 2022.

[67] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, and C. Manifavas, ‘A review of lightweight block ciphers’, *J. Cryptogr. Eng.*, vol. 8, no. 2, pp. 141–184, 2018.

[68] A. D. Dwivedi, P. Morawiecki, and G. Srivastava, ‘Differential Cryptanalysis of Round-Reduced SPECK Suitable for Internet of Things Devices’, *IEEE Access*, vol. 7, pp. 16476–16486, 2019, doi:

[69] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, "Simon and Speck: Block Ciphers for the Internet of Things", Available from: URL: https://eprint.iacr.org/2015/585.pdf.

[70] A. Alkamil and D. G. Perera, ‘Towards Dynamic and Partial Reconfigurable Hardware Architectures for Cryptographic Algorithms on Embedded Devices’, *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3043750.

[71] M. J. Dworkin, ‘Recommendation for block cipher modes of operation: The CMAC mode for authentication’, 2016.

[72] M. AbdulRaheem *et al.*, *"An Enhanced Lightweight Speck System for Cloud-Based Smart Healthcare"*, vol. 1455 CCIS. Springer International Publishing, 2021. doi: 10.1007/978-3-030-89654-6_26.

[73] M. H. Kashani, M. Madanipour, M. Nikravan, P. Asghari, and E. Mahdipour, ‘A systematic review of IoT in healthcare: Applications, techniques, and trends’, *J. Netw. Comput. Appl.*, vol. 192, p. 103164, 2021.

[74] T. Vaiyapuri, A. Binbusayyis, and V. Varadarajan, ‘Security, privacy and trust in IoMT enabled smart healthcare system: a systematic review of current and future trends’, *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 2, 2021.

[75] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and Y. Papaefstathiou, ‘A survey of lightweight stream ciphers for embedded systems’, *Secur. Commun. Networks*, vol. 9, no. 10, pp. 1226–1246, 2016.

[76] P. A. R. Patil and P. A. S. Kale, ‘Implementations of the Hummingbird Cryptographic Algorithm using FPGA’, pp. 104–109, 2015.

[77] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, ‘Hummingbird : Ultra-Lightweight Cryptography for Resource-Constrained Devices’, pp. 1–18.

[78] A. Haque, ‘Design of a hummingbird crypto asic implementing bist technique’, 2016.

[79] S. Saha, M. R. Islam, H. Rahman, M. Hassan, and A. B. M. A. Hossain, ‘Design and implementation of block cipher in hummingbird algorithm over FPGA’, *5th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2014*, pp. 2–6, 2014, doi: 10.1109/ICCCNT.2014.6963084.

[80] C. Oestreicher, ‘A history of chaos theory’, *Dialogues Clin. Neurosci.*, 2022.

[81] D. Levy, ‘Chaos theory and strategy: Theory, application, and managerial implications’, *Strateg. Manag. J.*, vol. 15, no. S2, pp. 167–178, 1994.

[82] I. Yasser, M. A. Mohamed, A. S. Samra, and F. Khalifa, ‘A chaotic-based encryption/decryption framework for secure multimedia communications’, *Entropy*, vol. 22, no. 11, pp. 1–23, 2020, doi: 10.3390/e22111253.

[83] L. Kocarev, ‘Chaos-based cryptography: a brief overview’, *IEEE Circuits Syst. Mag.*, vol. 1, no. 3, pp. 6–21, 2001.

[84] S. Amini and A. L. Steele, ‘A digital chaos generator for use within chaos encrypted communication systems’, in *2009 2nd International Workshop on Nonlinear Dynamics and Synchronization*, 2009, pp. 54–59.

[85] H. R. Dullin, S. Schmidt, P. H. Richter, and S. K. Grossmann, ‘Extended phase diagram of the Lorenz model’, *Int. J. Bifurc. Chaos*, vol. 17, no. 9, pp. 3013–3033, 2007, doi: 10.1142/S021812740701883X.

[86] R. Anandkumar and R. Kalpana, ‘analyzing of chaos based encryption with Lorenz and Henon map’, in *2018 2nd International Conference on I-*

SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2018 2nd International Conference on, 2018, pp. 204–208.

[87] L. Bin, L. Lichen, and Z. Jan, ‘Image encryption algorithm based on chaotic map and S-DES’, in *2010 2nd International Conference on Advanced Computer Control*, 2010, vol. 5, pp. 41–44.

[88] N. K. Pareek, V. Patidar, and K. K. Sud, ‘Image encryption using chaotic logistic map’, *Image Vis. Comput.*, vol. 24, no. 9, pp. 926–934, 2006.

[89] L. Bin, L. Lichen, and Z. Jan, ‘Image encryption algorithm based on chaotic map and S-DES’, in *2010 2nd International Conference on Advanced Computer Control*, 2010, vol. 5, pp. 41–44.

[90] P. R. Sankpal and P. A. Vijaya, ‘Image encryption using chaotic maps: A survey’, *Proc. - 2014 5th Int. Conf. Signal Image Process. ICSIP 2014*, pp. 102–107, 2014, doi: 10.1109/ICSIP.2014.80.

[91] H. Kezia and G. F. Sudha, ‘Encryption of digital video based on Lorenz Chaotic system’, *Proc. 2008 16th Int. Conf. Adv. Comput. Commun. ADCOM 2008*, pp. 40–45, 2008, doi: 10.1109/ADCOM.2008.4760425.

[92] R. Sobti and G. Geetha, ‘Cryptographic Hash functions - a review’, *IJCSI Int. J. Comput. Sci. Issues*, vol. 9, no. 2, pp. 461–479, 2012, [Online]. Available: https://www.researchgate.net/publication/267422045

[93] T. Krovetz, ‘UMAC: Message authentication code using universal hashing’, 2006.

[94] A. Sharma, ‘Comparative analysis of cryptographic hash functions’, *Cryptogr. hash Funct.*, 2018.

[95] M. Sumagita, I. Riadi, J. Sh, and U. Warungboto, ‘Analysis of secure hash algorithm (SHA) 512 for encryption process on web based application’, *Int. J. Cyber-Security Digit. Forensics*, vol. 7, no. 4, pp. 373–381, 2018.

[96] S. Al-Kuwari, J. H. Davenport, and R. J. Bradford, ‘Cryptographic hash functions: Recent design trends and security notions’, *Cryptol. ePrint Arch.*, 2011.

[97] A. Chowdhury and U. Kumar, ‘Performance Analysis of Keccak f-[1600]’, *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 8, pp. 236–241, 2013, doi:10.14569/ijacsa.2013.040832.

[98] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, ‘A statistical test suite for random and pseudorandom number generators for cryptographic applications’, Booz-allen and hamilton inc mclean va, 2001.

[99] C. C. Gan and G. Learmonth, ‘Comparing entropy with tests for randomness as a measure of complexity in time series’, *arXiv Prepr.*

arXiv1512.00725, 2015.

[100] C. Zhu, ‘A novel image encryption scheme based on improved hyperchaotic sequences’, *Opt. Commun.*, vol. 285, no. 1, pp. 29–37, 2012.

[101] Z. Hua, B. Zhou, and Y. Zhou, ‘Sine chaotification model for enhancing chaos and its hardware implementation’, *IEEE Trans. Ind. Electron.*, vol. 66, no. 2, pp. 1273–1284, 2018.

[102] M. Nadipally, ‘Optimization of methods for image-texture segmentation using ant colony optimization’, in *Intelligent data analysis for biomedical applications*, Elsevier, 2019, pp. 21–47.

[103] J. Rokan Naif, G. H. Abdul-majeed, and A. K. Farhan, ‘Internet of Things Security using New Chaotic System and Lightweight AES’, *J. Al-Qadisiyah Comput. Sci. Math.*, vol. 11, no. 2, pp. 45–52, 2019, doi: 10.29304/jqcm.2019.11.2.571.

[104] Z. M. J. Kubba and H. K. Hoomod, ‘Modified PRESENT Encryption algorithm based on new 5D Chaotic system’, in *IOP Conference Series: Materials Science and Engineering*, 2020, vol. 928, no. 3, p. 32023.

[105] C. E. Shannon, ‘A mathematical theory of communication’, *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

المستخلص

تُستخدم أجهزة إنترنت الأشياء على نطاق واسع اليوم ، وتتعامل هذه الأجهزة مع البيانات الرقمية مثل تدفق بيانات الفيديو. لأهمية هذه البيانات ، يجب إيجاد آلية أمنية متطورة لمنع انتهاكات خصوصية المعلومات ، حيث يجب تأمين كل جزء من هذه الأجهزة ، بالإضافة إلى WSN ، لمنع العديد من الهجمات.

في هذه الرسالة ، تم تصميم نظام تشفير لأنظمة المراقبة لحماية البيانات الرقمية المرسلة عبر شبكة إنترنت الأشياء. تعتمد الآلية المعتمدة للنظام على التشفير والمصادقة لتوفير الحماية للبيانات. تم تصميم آلية النظام المعتمدة وبنائها باستخدام خوارزميتين تشفير خفيفتين (الطائر الطنان والبقعة) وتم دمج الخوارزميتين معًا عن طريق التهجين بثلاث طرق مقترحة مختلفة. كل اقتراح عبارة عن خوارزمية تشفير هجينة خفيفة الوزن تتكون من أجزاء من خوارزمية البقع وأجزاء من خوارزمية الطائر الطنان ، ولكن بطريقة دمج مختلفة في كل طريقة. تم استخدام نظام Chaos أيضًا لإنشاء مفاتيح عشوائية للخوارزميات لجعل التشفير أكثر كفاءة.

تم استخدام خوارزمية التجزئة المعدلة الجديدة المقترحة أيضًا لمصادقة البيانات المشفرة لزيادة مستوى الأمان ولضمان مصداقية البيانات وموثوقيتها. تم تعديل نظام تجزئة SHA-3بسبب ضعف في SHA3 حيث يفقد السرعة المطلوبة لتحقيق تجزئة بطول ثابت ، مما يؤدي إلى إبطاء ويب الأشياء عند استخدام SHA-3 للتحقق من صحة المعلومات. لذلك ، تم اقتراح تصميم SFHASH3 من خلال دمج خوارزمية SPECK - FPHLE (تسع جولات لغرض تسريع عملية التشفير) كطبقة إضافية تكمل طبقات بنية الإسفنج الموجودة في.SHA3

أما بالنسبة لخوارزميات التشفير خفيفة الوزن المقترحة ، فقد أعطت نتائج تشفير مرضية ، واجتازت المقترحات الثلاثة جميع اختبارات .NIST تم حساب وقت التشفير وفك التشفير لكل خوارزمية هجينة مقترحة بأحجام إطارات مختلفة 128 × 128 ، 256 × 256 ، 512 × 512. بالنسبة لحجم الإطار 256 × 256 ، تم حساب وقت التشفير وفك التشفير بعدد مختلف من الجولات في كل مرة: 10 جولات ، 14 جولة ، 16 جولة. نتائج التشفير لأول خوارزمية خفيفة الوزن هجينة مقترحة لتشفير متوسط 7 إطارات ، وحجم الإطار 256 × 256 هو 16.024 مللي ثانية ووقت فك التشفير هو 15.927 مللي ثانية. قدم الاقتراح الثاني نتائج تشفير لنفس المواصفات السابقة: 33.81743 مللي ثانية للتشفير و 30.72014 مللي ثانية لفك التشفير. كانت نتائج الاقتراح الثالث 13.996 مللي ثانية للتشفير و 13.583 مللي ثانية لفك التشفير. كما هو موضح في النتائج ، فإن الاقتراح الثالث هو الأسرع ، لأن الخوارزميتين تعملان في نفس الوقت.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة بابل

كلية تكنلوجيا المعلومات – قسم البرمجيات

اقتراح خوارزمية تشفير لأمن تدفق الفيديو لأنترنت الاشياء في مراقبة المبنى

رسالة مقدمة الى

مجلس كلية تكنلوجيا المعلومات – جامعة بابل كجزء من متطلبات نيل درجة الماجستير في تكنلوجيا المعلومات / قسم البرمجيات

من قبل

نبأ علي خليل ابراهيم

بأشراف

أ.د حيدر كاظم حمود هاشم

1444هـ 2022م