ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
FACULTY OF INFORMATICS

THESIS

OE-NIK	Student's name:	**Mohammad Deyaa Al Din Fattal**
2022	Student's registration number:	**T/008795/FI12904/N**

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
FACULTY OF INFORMATICS

D AES: AN ENHANCED SECURITY APPROACH FOR IoT COMMUNICATION

Óbuda University
John von Neumann Faculty of Informatics
Institute for Cyber-physical Systems

DIPLOMA THESIS
TOPIC DESCRIPTION

Student name: **Mohammad Deyaa Al-Din Fattal**
Registration number: T/008795/FI12904/N
Neptun's code: [redacted]

Title of diploma thesis:

D advanced encryption standard: an enhanced security approach for IoT communication
D AES: Fokozott biztonsági megközelítés az IoT kommunikációhoz

Internal supervisor: Péter Fésüs
External supervisor

Deadline: 15th of May, 2022.

Topic description:

Design and implement a system that applies enhanced secured encryption algorithm D AES, that can be used in Internet of Things (IoT), ZigBee and similar applications. Apply the following:

- use symmetric key encryption where the sender uses the same key which the receiver uses,
- enhance the security in this type of encryption by making the keys nonlinear depending on the previous study,
- prevent knowing all sub-keys from attackers, because this algorithm has a strong structure,
- make more balanced between number of ones and zeros in cipher text,
- improve the key expansion function to get results more balanced compared to original design AES,
- make best Balance parameter between the number of ones and zeros in cipher text which is 50% percentage.

Simulate the developed algorithm D AES and analyse the performance.

The diploma thesis should cover:

- the precise description and detailed analysis of the problem to be solved,
- possible implementation methods and solutions,
- the detailed system plan and its justification,
- presentation of the technologies used in the implementation,
- a description of the implementation process, work phases and experiences,
- test methods, test plans and test results,
- the application and further development possibilities of the completed system,
- The source code of the developed system and the prepared documentation on a CD or DVD attachment.

.....
Dr. Rita Fleiner
head of institute

Term of limitation: **15th of May, 2024.**

The diploma thesis is adequate and can be submitted:

.....
external supervisor

.....
internal supervisor

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics

D AES: AN ENHANCED SECURITY APPROACH FOR IoT COMMUNICATION

by

Mohammad Deyaa Al-Din Fattal

Supervised by

Fésüs Péter

Assistant lecturer

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of computer science Engineering
May/2022

STUDENT'S DECLARATION

I the undersigned student hereby declare that this thesis is the result of my own work; I have disclosed the references and tools used in an identifiable manner. The results indicated in my thesis completed may be used by the university and the institution announcing the task for their own purposes free of charge.

Dated: Budapest, 01.05.2022

Mohammad Deyaa Al -Din Fattal

student's signature

CONSULTATION LOG
Thesis II (2020/2021)

Student's name:	Neptun code:	Specialty:
Mohammad Deyaa Al-Din Fattal	[redacted]	Master/computer science

Phone number: Mailing address (e.g. domicile):

[redacted]

Degree project / thesis[1] title in Hungarian:

D AES: Fokozott biztonsági megközelítés az IoT kommunikációhoz.

Degree project / thesis[2] title in English:

D advanced encryption standard: an enhanced security approach for IoT communication.

Institutional supervisor: External supervisor:

Fésüs Péter .....

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	2021-10-12	Tasks for the semester	Fésüs Péter
2.	2021-10-26	Review 1	Fésüs Péter
3.	2021-11-23	Review 2	Fésüs Péter
4.	2021-12-07	Final review, presentation	Fésüs Péter

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, 13/12/ 2021

Fésüs Péter
.....
institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

CONSULTATION LOG
Thesis III (2021/2022)

Student's name:	Neptun code:	Specialty:
Mohammad Deyaa Al-Din Fattal	[redacted]	Master/computer science

Phone number: Mailing address (e.g. domicile):

[redacted]

Degree project / thesis[1] title in Hungarian:

D AES: Fokozott biztonsági megközelítés az IoT kommunikációhoz.

Degree project / thesis[2] title in English:

D advanced encryption standard: an enhanced security approach for IoT communication.

Institutional supervisor: External supervisor:

Fésüs Péter .....

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	2022-01-10	Tasks for the semester	
2.	2022-01-20	Review 1	
3.	2022-02-15	Review 2	
4.	2022-02-26	Final review, presentation	

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, 02/03/ 2022

.....
institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

CONSULTATION LOG
Thesis IV (2021/2022)

Student's name:	Neptun code:	Specialty:
Mohammad Deyaa Al-Din Fattal	[redacted]	Master/computer science

Phone number: Mailing address (e.g. domicile):

[redacted]

Degree project / thesis[1] title in Hungarian:

D AES: Fokozott biztonsági megközelítés az IoT kommunikációhoz.

Degree project / thesis[2] title in English:

D advanced encryption standard: an enhanced security approach for IoT communication.

Institutional supervisor: External supervisor:

Fésüs Péter .....

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	2022-03-10	Tasks for the semester	Fésüs Péter
2.	2022-03-22	Review 1	Fésüs Péter
3.	2022-04-15	Review 2	Fésüs Péter
4.	2022-04-27	Final review, presentation	Fésüs Péter

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, 01/05/ 2022

Fésüs Péter
.....
institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

Dedication

I dedicate my thesis work to my family and my friends. A special feeling of gratitude to my loving mom Hiba, my father Hisham, My brother Ahmed, My sisters Zahraa and Aya that have never left my side and always supported me.

Acknowledgment

First and foremost, I want to express my gratitude to Almighty Allah for bestowing His blessings on me and providing me with the strength to carry out and complete this work.
I am extremely grateful to my supervisor **Mr. Péter Fésüs** for his valuable advice, guidance, beneficial discussions and encouragement throughout my research. Apart from his valuable academic advice and guidelines, he has been extremely kind, friendly, and helpful.
I'd like to express my gratitude to my parents, brother, and sisters for their unwavering support, patience, and love. It would have been impossible for me to complete this work without their support, motivation, and understanding. A special thanks for my friend Sadek for his kind support and motivation. Finally, I want to express my gratitude to my faculty and my university "**John von Neumann Faculty of Informatics | Óbuda University**" and to all my professors and staff and everyone who supported me to complete this project during these two years.

Abstract

With the rapid development of internet-connected technologies and applications, people are keen to embrace the convenience and practical applications they bring. With the improvement of all kinds of technologies, the Internet of Things is maturing and being able to provide more advanced services to people, which connect a variety of devices, systems, and applications other than the traditional device and the machine. The continued growth of internet-connected devices provides exciting opportunities for future technology. These Internet of Things (IoT) products are manufactured and networked with many daily activities, creating a greater security concern. The sensors will collect previously unimaginable amounts of private and public data and transmit it all over an easily observable wireless medium so that other devices can perform data analyzes.

Ensuring the security of the sanative data and traffic of the Internet of Things from being accessed by unauthorized user is very important issue, especially while being transmitted or stored for the companies and end users.

Multiple ways are used to do this job, one of the most famous is to use cryptography. Cryptography is used to transfer the data in a form that is not understood by anyone apart from the intended recipient. Many standards and developed encryption protocols are available as resources and are used based on the requirements. The advance standard encryption algorithm (AES) is one of the most secure encryption algorithms, but AES suffers from consumption of unnecessary time to achieve the necessary complexity needed to meet the security level, especially for real time application. In this thesis, we propose an enhanced encryption algorithm to safely transfer information. [1]

AES is considered as synchronize encryption whose key are the same in the sender and the receiver. All the research work to enhance the security in this type of encryption by make the keys nonlinear. So, if attackers know any one of sub-keys this will lead to know other keys in not difficult way. To make it more nonlinear and more difficult to be attacked it should be more balanced between number of ones and zeros in cipher text so we make some changes in key expansion function to get results more balanced compared to original design AES.

Contents

1 Chapter 1: Introduction and the Aim of Research

1.1 Background

Information security has become a matter of utmost importance, especially after the introduction of computers into all aspects of life. The use of networks and means of communication to transfer data, and confidentiality of information is considered.

The primary goal of encryption techniques and algorithms is where information is protected and rendered incomprehensible or illegible. Except with prior permission and authority consistent with the encryption technology used, and so the exchange of confidential information.

The high level between the two parties to the communication will not enable any intrusive party to obtain this information, as well as the rules Data protected by confidentiality security technologies will only be accessed by authorized and authorized persons.

1.1.1 Internet of Things devices and protocols

Devices with restricted resources, such as CPU, memory (ROM and RAM), and battery life, are known as constraint devices. Sensors, machine to machine (M2M) connectivity, and smart devices managing electrical appliances or services are all common uses for devices. When these devices are connected to the internet, they are referred to as "things" and become part of the "internet of things." IoT is a network of things with the ability to connect and exchange data with other devices and services, such as embedded computers, controllable and intelligent automated equipment (smart devices), and sensors. Home automation, manufacturing, environmental monitoring, medical and health-care systems, and transportation are all examples of IoT applications. IoT devices might potentially give information and services in the physical world to everyone, anytime, and everywhere if they are connected to the internet. [3] IoT allows users to obtain information about their devices that has been uploaded and saved on a web server or cloud storage, as well as interact with and control their devices via web, mobile, and cloud interfaces and applications. [2]

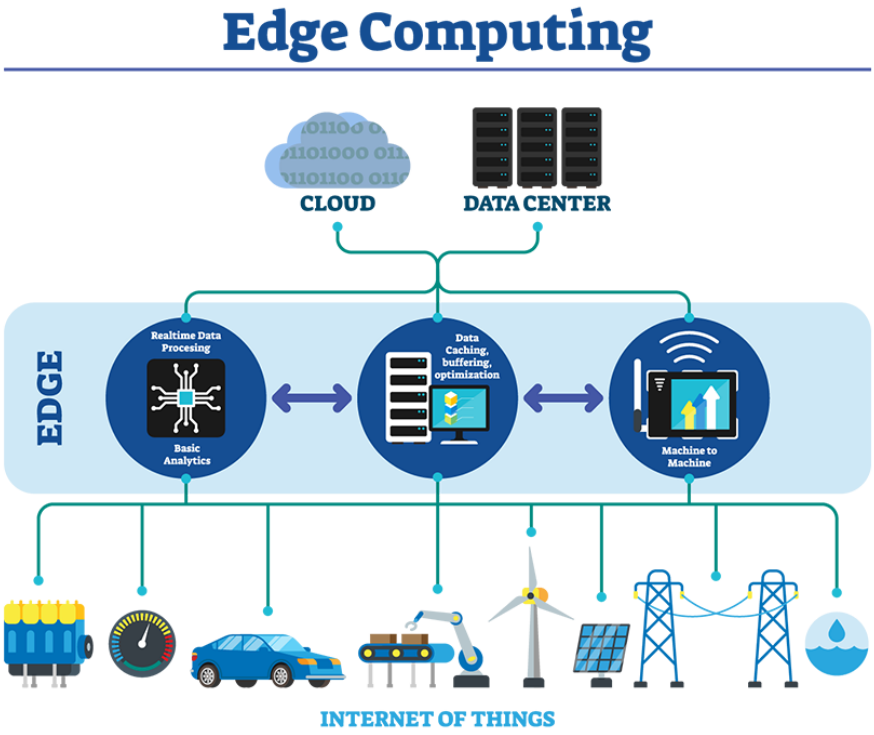

Figure 1-1: IoT deployment and application scenarios.

1.1.2 Security in IoT devices

Security is described in this thesis as the safeguarding of data from unauthorized access or surveillance by assuring data confidentiality, integrity, and authenticity. The protection of data against disclosure to unauthorized persons, parties, or systems is known as data confidentiality.

The prevention of data falsification or change by unauthorized parties is characterized as integrity. The verification of a device's or system's identification is referred to as authenticity.

Since 1990, the Hypertext Transfer Protocol (HTTP) has been used for data exchange on the World Wide Web as an application layer protocol for distributed, collaborative, hypermedia information systems. HTTP is deemed insecure as a standalone protocol since it transfers data in plaintext and does not use security features to protect data. A more secure solution was required as the amount of sensitive data exchanged over the internet increased. As a result, HTTP over Secure Socket Layer (SSL) and its successor Transport Layer Security were developed (TLS). This combination produced HTTPS, a secure communication protocol that uses cryptography to prevent eavesdropping, tampering, or message forgery. HTTP and HTTPS use the Transport Layer

Protocol Transmission Control Protocol (TCP), which ensures data transmission dependability, error protection, and flow control. To ensure that the transmission is reliable, these data verification functions necessitate additional system resources. Because HTTP and HTTPS were not designed for IoT devices with limited resources, a more efficient protocol was created specifically for these devices. The Constraint Application Protocol (CoAP) is a resource-constrained application layer protocol designed for IoT devices. CoAP, like HTTP, has REST techniques and was created so that the two protocols could readily communicate with one another. Unlike HTTP, which uses TCP, CoAP uses UDP by default, which is a less complex protocol that requires less header information than TCP, making it more suitable for devices with limited resources. Because UDP does not check data transfer for mistakes by default, it is regarded as an unreliable protocol. Because of the confined nature of IoT, devices have limited resources and can only handle a limited number of protocols and processes. While these devices may have restricted capability, they may nevertheless be able to gather and communicate personal sensitive information to online or cloud services over the internet. UDP is regarded as an unstable protocol because it does not check data transport for errors by default. Because of the restricted resources available in IoT, devices can only handle a limited number of protocols and operations. Despite their limited capabilities, these devices may be able to collect and transmit personal sensitive information to online or cloud services through the internet. In the same manner that TLS secures HTTP communication on the web (HTTPS), DTLS protects data confidentiality, integrity, and authenticity of CoAP connections. While DTLS is suitable for some IoT devices, it is still a resource-intensive protocol, and devices must have enough resources to operate it while still performing their intended job, such as collecting data from a temperature sensor. [2]

1.1.3 Cryptography in IoT

TLS (Transport Layer Security) uses cryptography to ensure a secure and trustworthy data communication connection. The sender encrypts data using the recipient's cryptographic public key. The data is subsequently transferred to the recipient over the internet. Only the recipient's private key can decode the material, and the recipient keeps this key private and secure. TLS is also used to establish a secure session by securing the exchange of symmetric keys, which are subsequently utilized for the majority of the data exchange. Because a security handshake and key exchange must occur, asymmetric cryptography methods require more resources to function than

symmetric encryption methods. Asymmetric cryptography and (D)TLS protocols are too resource intensive for Class-0 devices due to their restricted resources.

While asymmetric cryptography may be too resource-intensive to secure communication in Class-0 devices, symmetric encryption provides a low-resource option. Symmetric cryptography encrypts data with a single encryption key that can be used by different devices. Any device with the key can decrypt data sent by other devices with the key. Because the key is shared, the chance of it falling into the wrong hands is higher, so it must be kept safe. When more than two or more IoT devices share the same key, they form a group and are verified as a group rather than as individual devices. The Advanced Encryption Standard (AES) is a symmetric standard that operates at high speeds and uses fewer resources than (D)TLS, making it ideal for Class-0 devices. AES accepts data in 16-byte (128-bit) blocks, which are subsequently encrypted with a 128-bit, 192-bit, or 256-bit cryptographic key. The bigger the key, the more security and resources the device needs to encrypt and decrypt data. Symmetric encryption can be used at several tiers of the communication stack, including the data link layer (for example, wireless transmissions) and to specific data objects inside a message, such as sensor readings. [2]

1.2 Aims and Objectives

There are two types of encryptions; in the first type which called synchronize encryption, the sender uses the same key which the receiver uses.in Asymmetric encryption private key and public key are used from the sender and the receiver. AES is symmetric key encryption. In some of applications related to security, AES algorithm is used like ZigBee, Internet of things (IOT) and other applications.

The structure of AES is considered strong, but it can be enhanced to make it more non-linear to prevent knowing all sub-keys if any one of them are known from attackers. Balance parameter depends on number of ones and zeros in cipher text. The best percentage is 50% which means 64 ones and 64 zeros in the cipher text, and it is the most non-linear percentage. The proposed model for enhanced algorithm is D AES, it represents the first letter of the author’s name.

1.3 Outlines of the Thesis

The thesis ‘D ADVANCED ENCRYPTION STANDARD: AN ENHANCED SECURITY APPROACH FOR IoT COMMUNICATION’ is divided into six Chapters are as follows:

1 The first chapter introduces the research and explains the importance of security in the Internet of things, the objectives of the research, and a brief explanation for thesis chapters.

2 The second chapter introduces a literature review on the basic introduction to cryptography and presents the detailed explanation of the AES algorithm, which is needed to understand the proposed approach.

3 The third chapter include the previous work and studies, it presents the reference studies for some modifications and improvements of the advanced encryption standard algorithm and the results of each of them.

4 The fourth chapter includes the Proposed Mechanism for Improving AES Algorithm and introduces the methodology of the customized encryption approach. With the implementation of DAES using MATLAB software.

5 The fifth chapter includes a summary of the results reached in this Research, and the comparison with original algorithm.

2 Chapter 2: Cryptography and AES Algorithm

2.1 Introduction to Cryptography

In the modern systems, cryptography deals with a lot of problems. But the basic one is to ensure the security of information in a communication channel. For demonstration, let us assume that there are two communication sides the sender which will be called A and the receiver which will be called B, and they want to communicate securely with each other as shown in Figure 2-1: Secure link between Alice and Bop.[4]

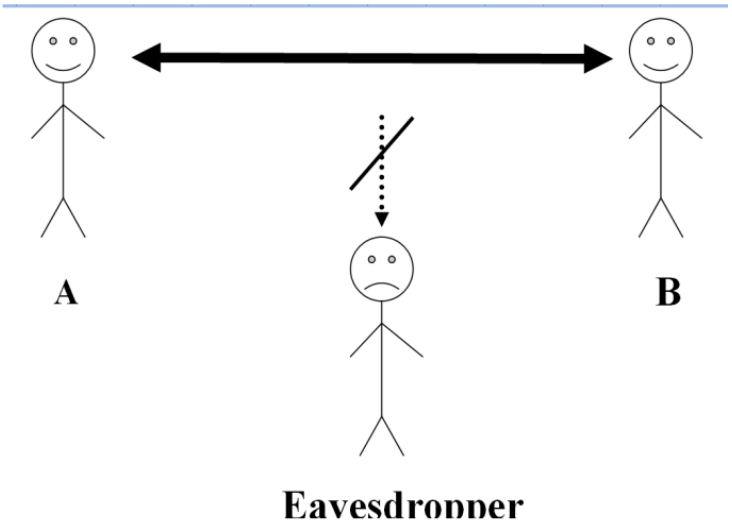

Figure 2-1: Secure link between Alice and Bop.

The most basic aim for Cryptography is to provide an ideal channel between A and B so over an insecure channel so no one such as eavesdropper can listen to the transmitted data between A and B. But in general, the cryptography provides two goals:

1- Privacy: hiding the content of a transmission form an eavesdropper.
2- Authenticity or Integrity: ensuring that the receiver has the message from the predetermined transmitter and prevent any eavesdropper from taking the receiver or transmitter identity.

To achieve the security goals such privacy or authenticity, Cryptography distributes a protocol that contains software and rules to each party in the secure link of communication that does not leak

any information they don't want to be known. The software contains the sender algorithms that will secure the data and ensures the security goals before sending it over the insecure channel and the receiver algorithm that let the receiver accepts the data or denies it when it has security errors such as eavesdropper modification on the data.

All cryptography protocols depend on a Key, which is shared between the parties of communication, and the cryptography algorithms are divided into two types: Symmetric and Asymmetric Encryption Algorithm.

2.2 Symmetric Cryptography

Sometimes, it called pre-shared key. In the symmetric encryption algorithm, the key between the sender and the receiver is the same key and this key must be distributed to all parties involved in the secure link to ensure that the eavesdropper does not have it.

In this kind of encryption, the encryption process and decryption process are done using the same key as shown in Figure 2-2: The symmetric algorithm uses the same key for encryption and decryption. [4]

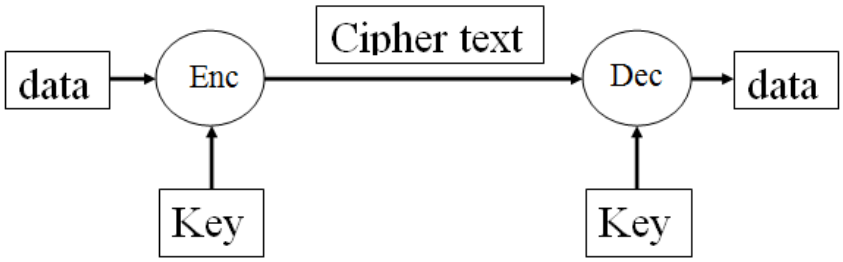

Figure 2-2: The symmetric algorithm uses the same key for encryption and decryption.

2.3 Asymmetric Cryptography

Sometimes it called public-key algorithm. In the asymmetric encryption algorithm, the key is divided into two pieces: the first is called the private key and the other called the public key. In order to make secure link between the sender and the receiver the sender must have the public key and it can be sent over an insecure channel because it is used only for the encryption of the data,

and it cannot be used to decrypt it. Meanwhile, the receiver which has the private key will be able to decrypt the data using it and the private key must be stored securely. [4]

The process of encryption and decryption process for this kind of algorithm is shown in Figure 2-3: The Asymmetric algorithm uses different keys for encryption and decryption (Gilbert, 2017).

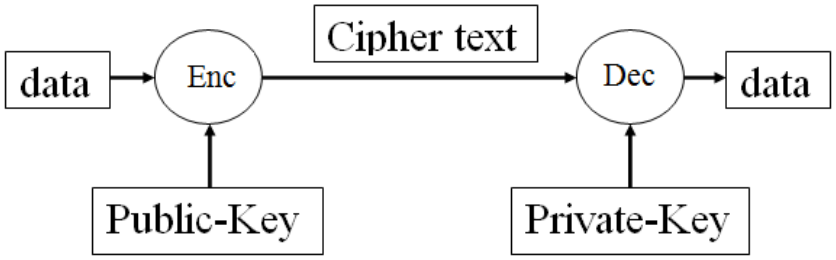

Figure 2-3: The Asymmetric algorithm uses different keys for encryption and decryption.

2.4 Advanced Encryption Standard (AES)

2.4.1 Introduction

Advanced Encryption Standard (AES) is published by the National institute of Standards and Technology NIST in 2001, which is one of the properties of symmetric block encoding algorithms. Whereas the encryption and decryption process is done using the same encryption key.

2.4.2 AES Bytes and Words

The one byte and one word are represented as a matrix as shown in Figure 2-4: Matrix representation of bytes and words in AES algorithm. While one block (128 bits) is divided into four words, and the formation of the status matrix shown in Figure 2-5: Forming the state matrix in the AES algorithm from the 128-bit block.

Each word (every four bytes) belongs to a column in the state matrix.so forming the state matrix from the block and extracting the block from the state matrix is accomplished according to the mechanism shown in Figure 2-6: Forming the state matrix from the block and extract the block from the state.[5]

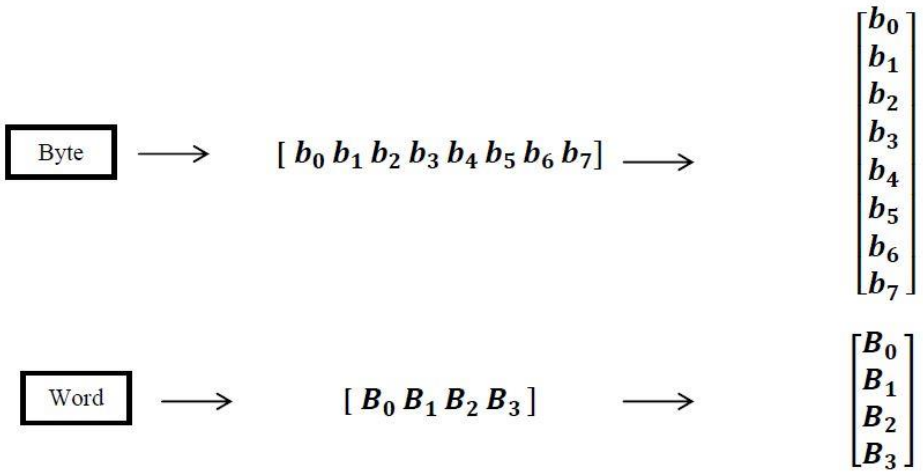

Figure 2-4: Matrix representation of bytes and words in AES algorithm.

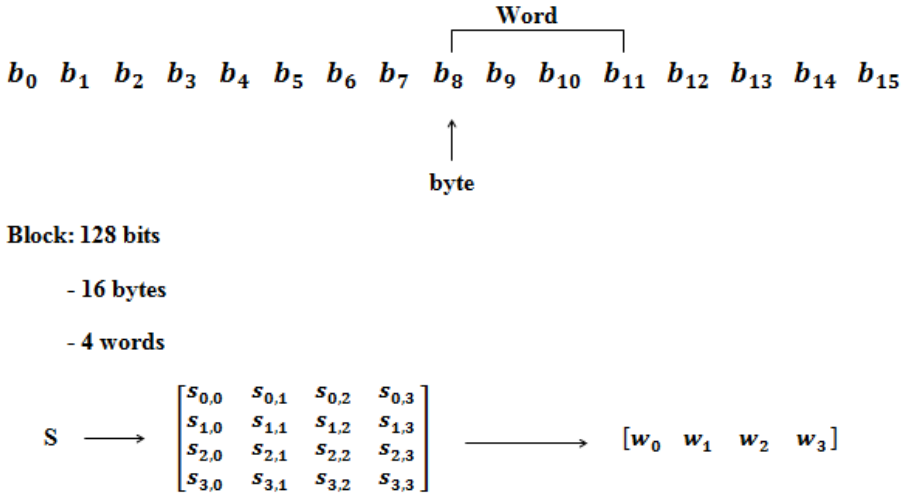

Figure 2-5: Forming the state matrix in the AES algorithm from the 128-bit block.

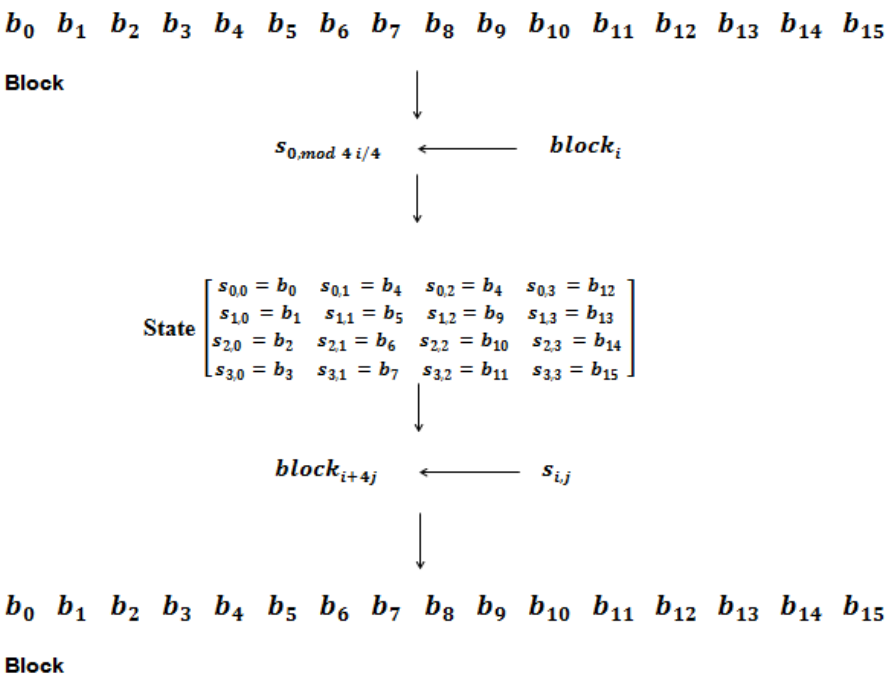

Figure 2-6: Forming the state matrix from the block and extract the block from the state.

2.4.3 AES General Structure

The Figure 2-7: AES General Structure shows the general structure of AES algorithm. Where the input length is 128 bits, and the number of algorithm rounds depends on key size as shown in the Table 2-1: the relation between key size and number of rounds in AES algorithm.[6]

Indicting to the algorithm according to the length of the encryption key that been used. AES-128, AES-192, AES-256.

Key size	Number of rounds
128	10
192	12
256	14

Table 2-1: the relation between key size and number of rounds in AES algorithm

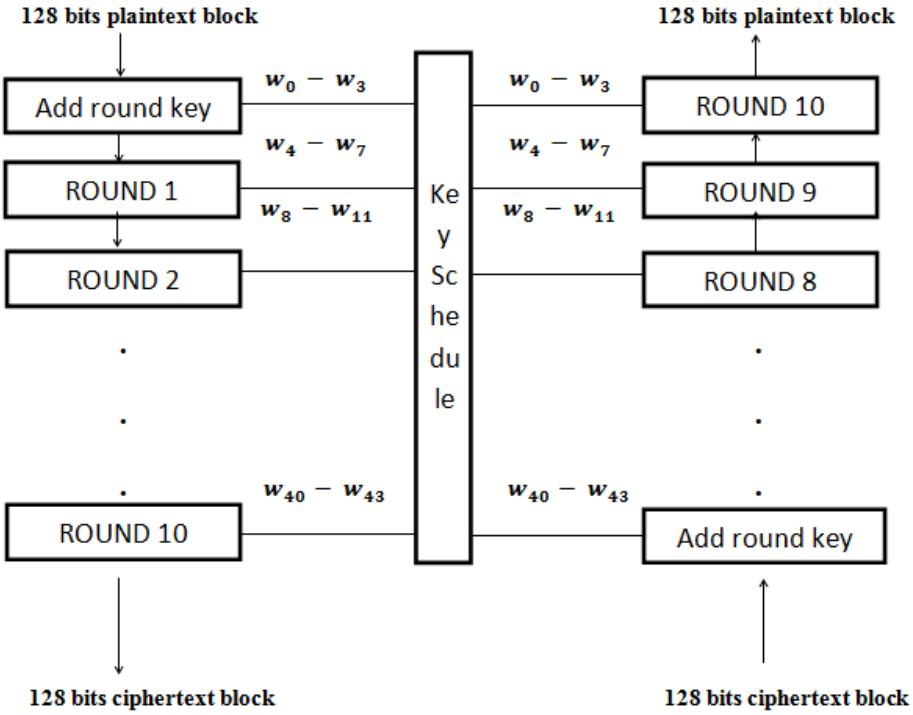

Figure 2-7: AES General Structure.

The sub keys for algorithm rounds are obtained after applying the primary key to key expansion mechanism. So, the algorithm's input will be the state matrix that result from the input block and the primary key that will be presented also as a matrix.

Let's see the detailed AES structure for 10 rounds and key size=128 bits as shown in the Figure 2-8: AES Detailed Structure.[6]

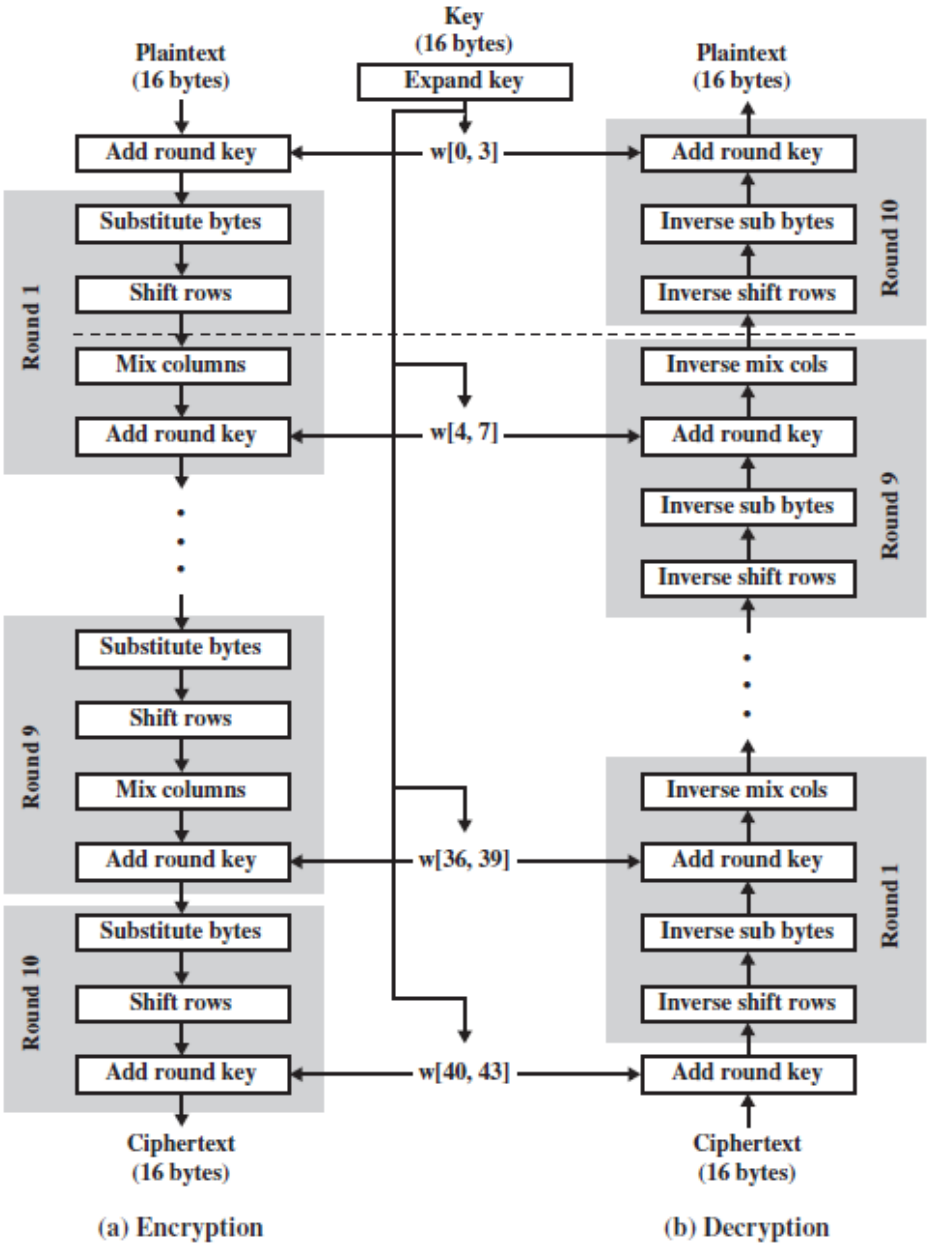

Figure 2-8: AES Detailed Structure.

2.4.4 AES One Round Structure

The initial stage contains adding the encryption key to the plain text using binary XOR operation, then executing the algorithm loops. Where single round contains of the following stages as shown in the Figure 2-9: AES One Round Structure.[6]

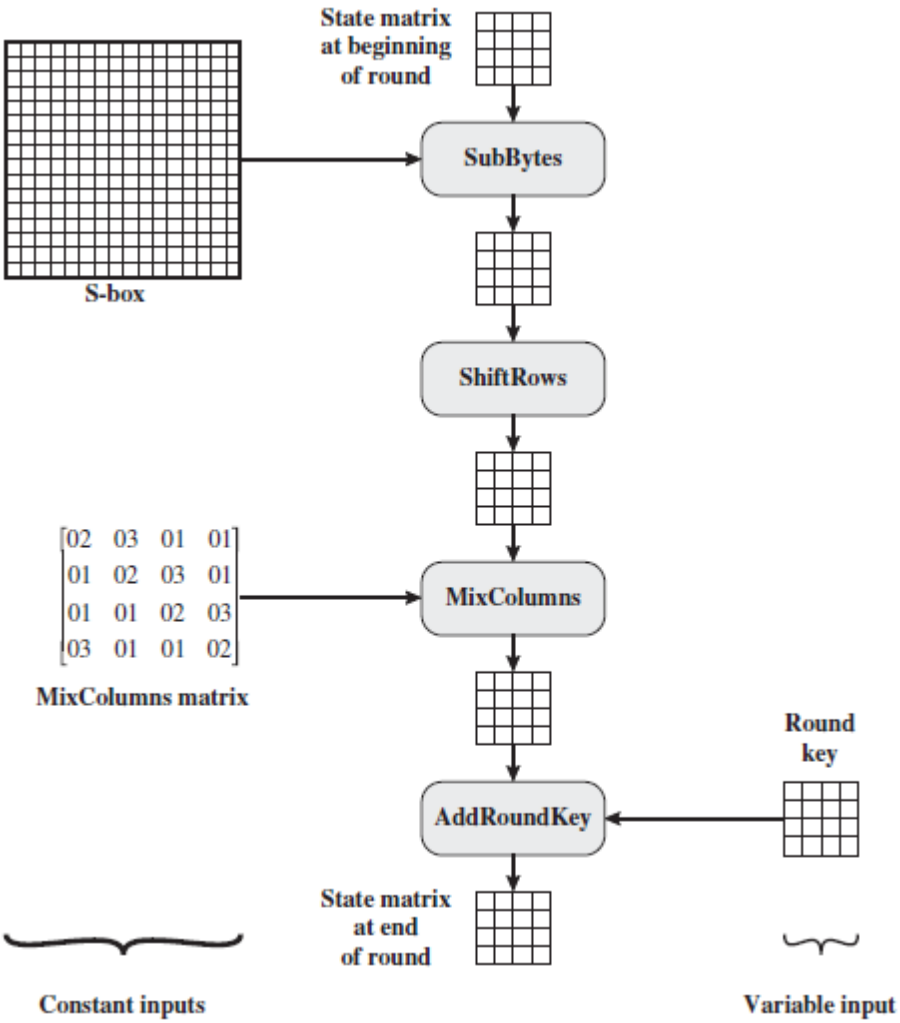

Figure 2-9: AES One Round Structure.

2.4.4.1 The Sub-Bytes Step

The resulting state matrix 'bytes from the previous stage are replaced by using the S-Box as shown in the Figure 2-10: AES One Round Sub-Bytes Step, where Byte substitution is applied on all state matrix elements by replacing its elements values by S-Box elements values. By taking the index of each element (the first 4-bits on the left hand determine the line value and the second 4-bits determine the column value in the S-Box). And then we get the alternative value resulting from crossing the line and column in S-Box.[6]

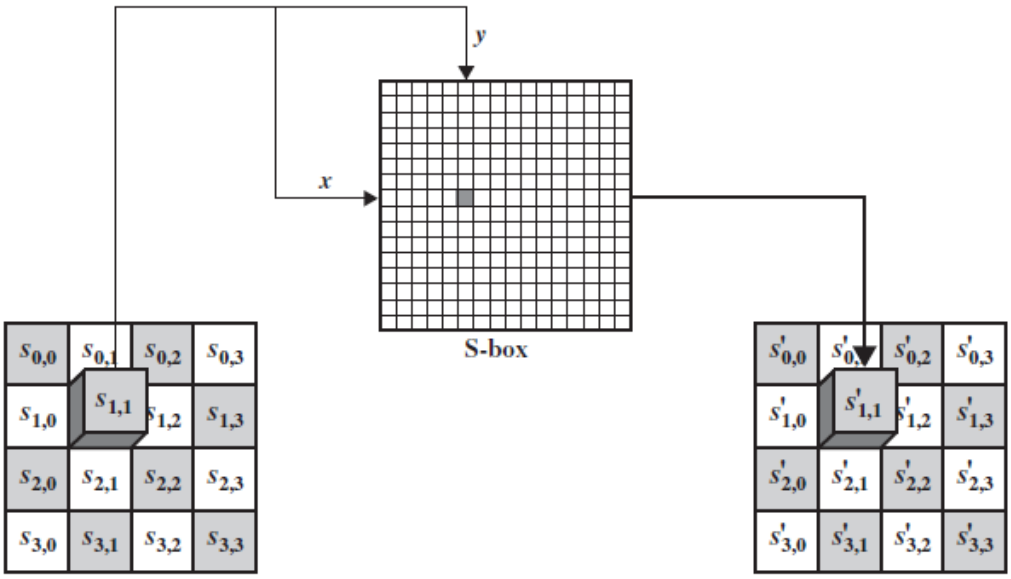

Figure 2-10: AES One Round Sub-Bytes Step.

Where we have a special **S-box** used in the encryption process and a special **Inverse S-box** used in the decryption process. Each box has a 16 x 16-byte matrix containing permutations of up to 256 possible values for 8 bits, and those values are created in a specific order, and Figure 2-11: S-Box and Inverse S-Box values (a) shows the S-box values and Figure 2-11: S-Box and Inverse S-Box values (b) show the Inverse box values.[5]

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

(a) S-box

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

(b) Inverse S-box

Figure 2-11: S-Box and Inverse S-Box values.

2.4.4.2 The Shift-Rows Step

This process is carried out as shown in Figure 2-12: AES One Round Shift-Rows Step, where the offset of lines in the Encryption process is done according to the following:

The first line of the state matrix remains without offset, and the second line is shifted one position to the left. The third line is shifted two positions to the left while the fourth line is shifted three positions to the left. In the decryption process, the same process is performed, but instead of the circular shifting to the left, the circular shifting is to the right.[5]

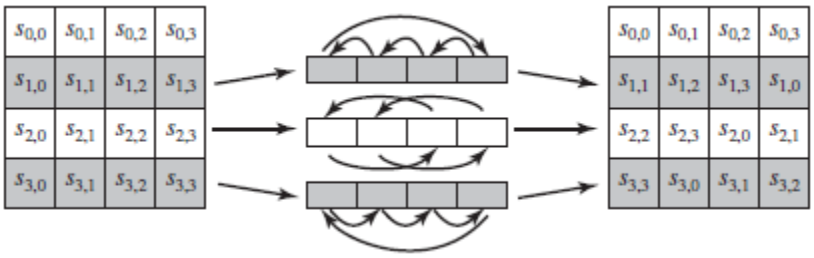

Figure 2-12: AES One Round Shift-Rows Step.

2.4.4.3 The Mix-Columns Step

This process is performed by multiplying each column of the resulting state matrix from the previous step by a square matrix with the same dimensions of the state matrix.

The Figure 2-13: AES Encryption Mix-Columns Step shows the process of mixing columns in the encryption process, whereas the Figure 2-14: AES decryption Inverse Mix-Columns Step shows the process of inverse mixing columns in the decryption process.[6]

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Figure 2-13: AES Encryption Mix-Columns Step

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Figure 2-14: AES decryption Inverse Mix-Columns Step.

2.4.4.4 Add Round Key

This process is performed by applying the binary operation XOR on the 128-bit state matrix with the 128-bit private key of each round.

Where applied XOR on each four bytes of state matrix with one WORD of keys expansion matrix.

The use of this stage affects every bit in the state array, so the more complex the extended key, the more complex the other stages of the AES algorithm will be. And that leads to Increasing the degree of safety.[8]

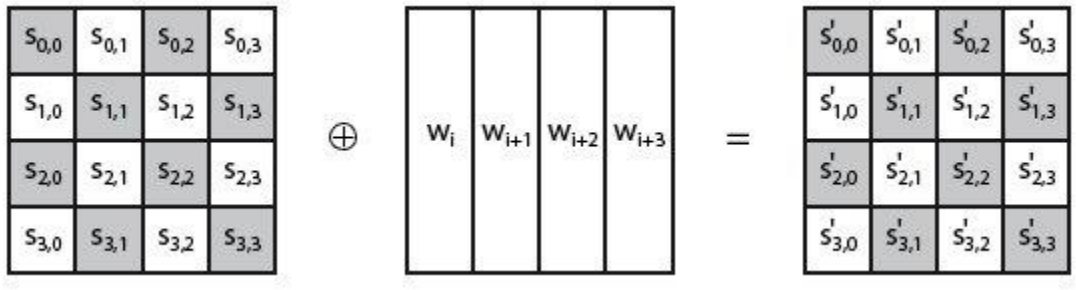

Figure 2-15: Add Round Key Step.

2.4.4.5 Expansion

The encryption key is applied to the input of the key expansion algorithm as a square matrix consists of 4 words 16 byte. Resulting in a linear matrix of 44 words (11 keys), i.e. the key to the initial round plus ten Keys. In this algorithm to find the extended key W[i] a more complex function is used. We have the round constant Rc [j] has different values for each round and these values are shown in Table 2-2: the values of the round constant RC[j] for each round..

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

Table 2-2: the values of the round constant RC[j] for each round.

The key expansion algorithm is performed according to the following steps:

1- Shifting 1 byte circularly to the left on the word W[i-1] in the previous key.
2- Replacing each byte of the previous word with its corresponding S-box.

3- The result is summed with the round constant Rc [j] with the word W[i-4] according to the binary operation XOR, and that constant is different for each round, and is calculated based on the powers of the number (x=02), which are values according to Hexadecimal representation.

The 4-Word (16-byte) are applied to the input of key expansion algorithm, which is the Basic secret key resulting a linear matrix of words to provide the 4-Word key for the initial round as well as to provide all the round keys in the algorithm.[5]

The expansion process is performed by following:

1- Putting the primary key into an matrix with four words as the state matrix.
2- Creating groups of four consecutive words derived from the previous values, and the previous values from the four words with using the **g** function which represents multiples of 4.
3- The first word of each group has a special processing (cycle shifting, S-box, XOR) as shown in Figure 2-16: The main steps of the key expansion algorithm in AES considering in case the key is 256 bits/14 round there an extra step on the middle word. [7]

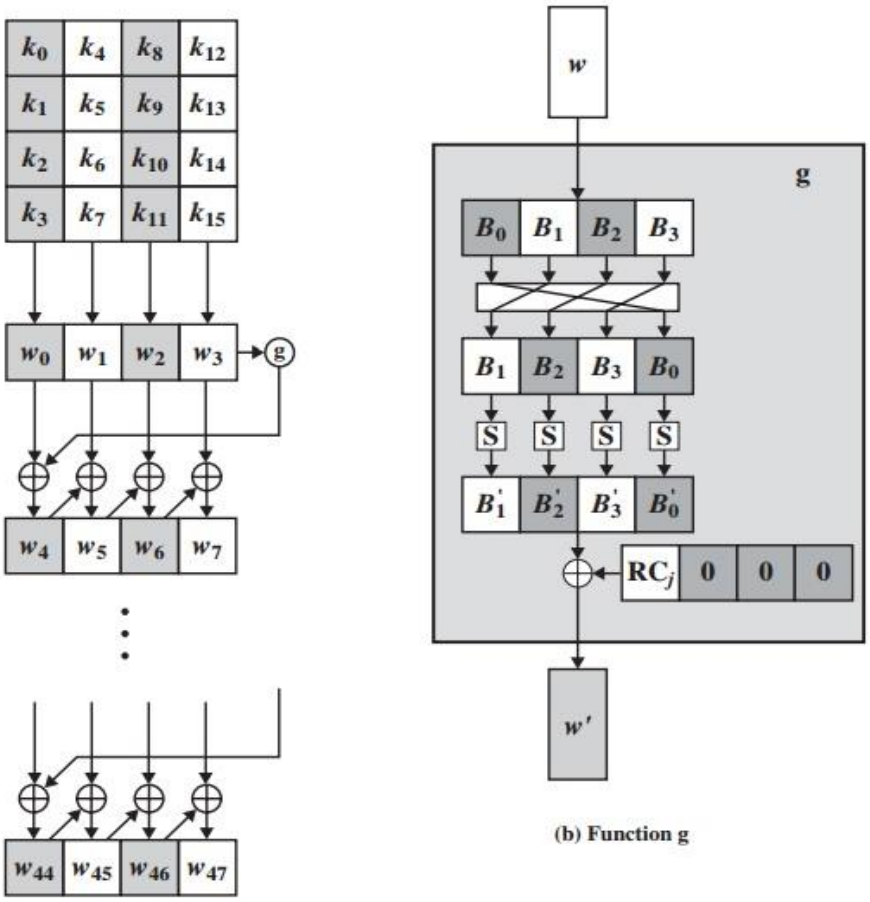

Figure 2-16: The main steps of the key expansion algorithm in AES.

3 Chapter 3: Previous work and reference studies

3.1 Introduction

This chapter provides reference studies of some techniques for improving the performance of AES encryption algorithms.

In some of this techniques the researchers concentrate on how to improve the speed of this algorithms, and the others concentrate on how to increase the security of this algorithm.

3.2 Previous improvements to the AES encryption algorithm

In many of related papers, researchers have presented many ways to improve the performance of encryption algorithms AES.

In reference [9] the researchers presented a method for increasing the performance of the AES encryption algorithm, by replacing the Shift rows process with the indexing process where this circular process can be faster using a layout of the shift rows matrix that contains an index value that is set directly to the state matrix index as shown in the Table 3-1: shift rows matrix layout for indexing process.

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
value	0	5	10	15	4	9	14	3	8	13	2	7	12	1	6	11

Table 3-1: shift rows matrix layout for indexing process.

They also combined the two phases (Sub-bytes – Mix-columns) in one stage which are executed within each round, reducing the execution time of the algorithm by a rate of 86.134 % for the encryption process and a rate of 13.085 % for the decryption process, but this method did not address the security. Also, it needs a large memory to store the layout of the shift rows matrix for the Indexing process.it also requires high computer specifications.

In the reference [10] the researchers developed the AES encryption algorithm by increasing the size of the input block to become 512-bit and, they used a 512-bit key, which made the algorithm

more confidential and more productive. They called it High Lightweight Encryption Standard “HLES”.

They relied on dividing the input and the used key into four blocks, each one with a length of 128 bits. And then this algorithm was implemented with four consecutive rounds, each round has input of 128 bit and a key length of 128 bit. Where the first round consists of three steps (Sub Bytes Substitution, Zero shifting SH-Z, Add Round Key). Whereas the next three rounds are related to the output of the previous round for each one, each round consists of three steps (the XOR process with the output of the previous stage L-XOR, the Zero shifting SH-Z, Add Round Key). This reduces the size t of used memory, and reduces the execution time, where the researchers compared this algorithm with the traditional AES algorithm and the AES-512 algorithm by applying the same text, audio, and image files to each of these methods and they found that the proposed algorithm is the best in terms of execution time and the size used memory.

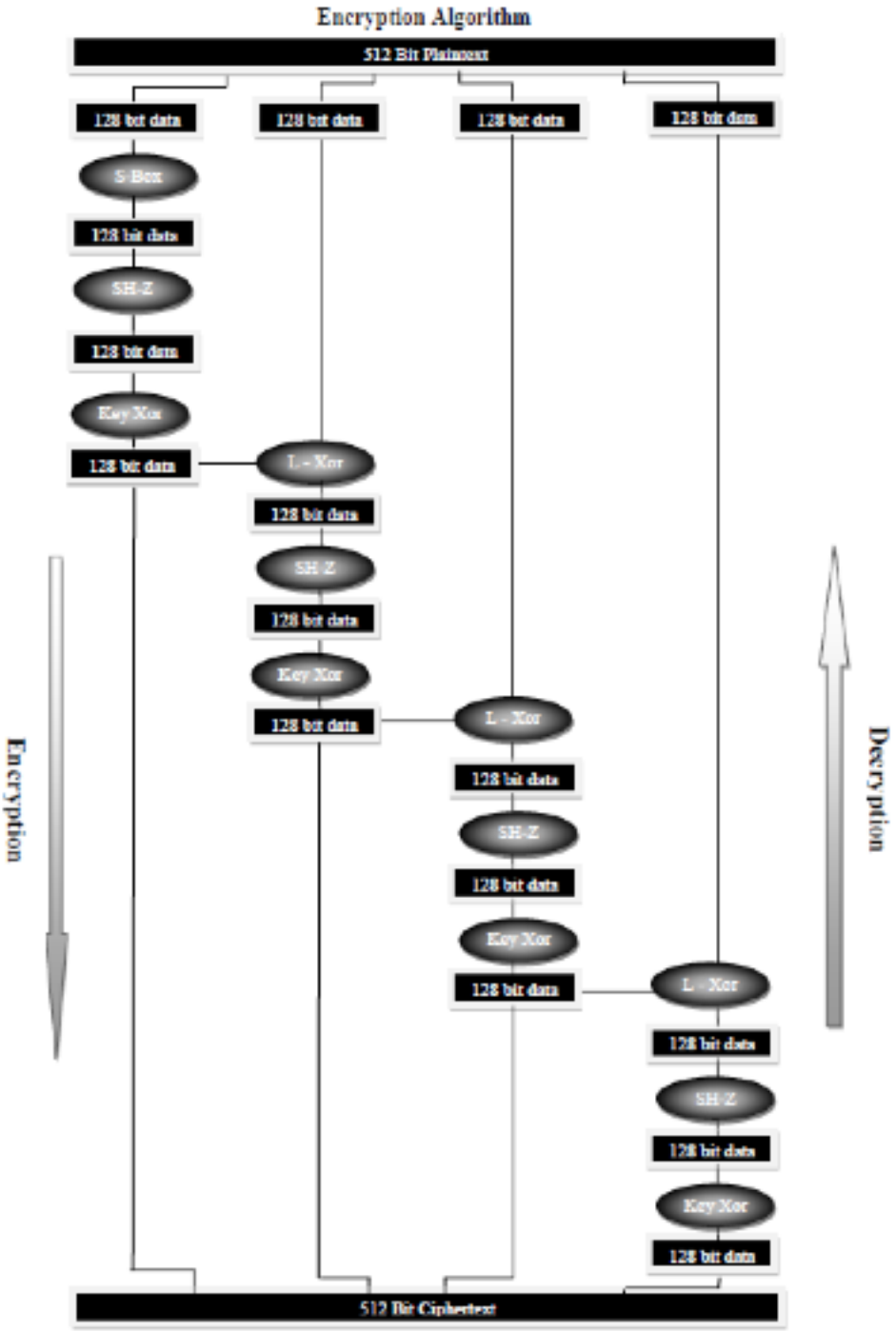

Figure 3-1: HLES General Structure.

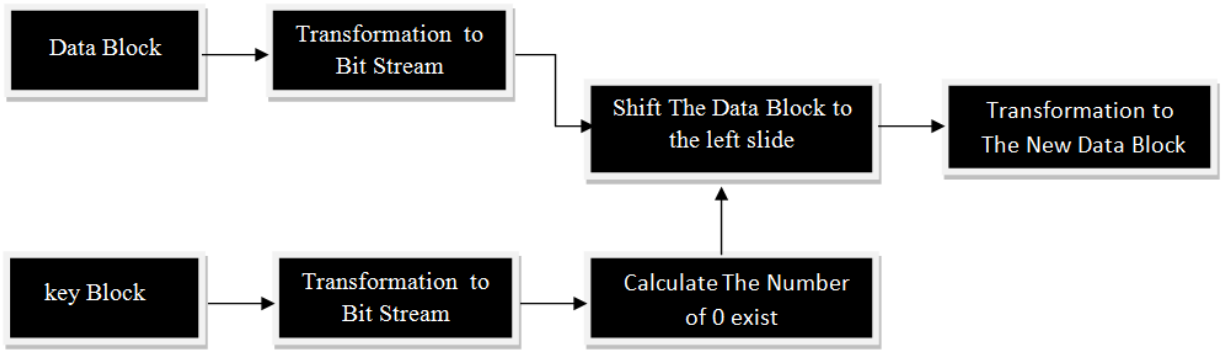

Figure 3-2: Zero shifting SH-Z process.

In the reference [11] the researchers have modified the key expansion in in the g function step, where they replaced the zeros fields with words of the primary key.

The round content RC[j] values will be as it is. And the other filled will be filled with one of key's words. As shown in the following Figure 3-3: Replacing zeros fields with words.

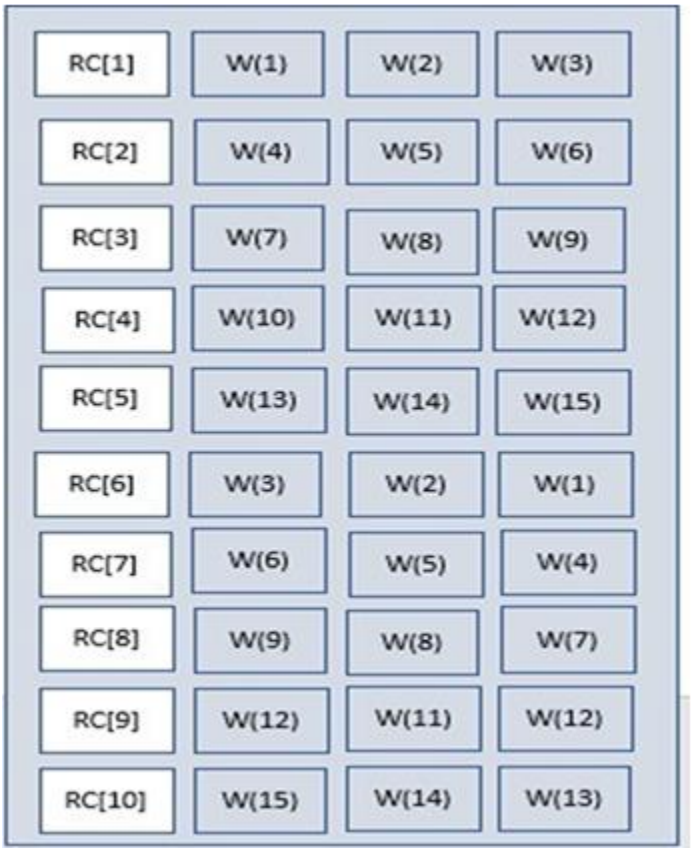

Figure 3-3: Replacing zeros fields with words.

The researchers take this pattern of words distribution cause it is leads to best results of balance parameters between zeroes and ones.

In the reference [12] the researchers reduced the run time to execute the algorithm by reducing the overhead calculations in both encryption and decryption side. Their purpose was to increase the algorithm's speed. And they achieve that by replacing the mix –column stage with permutation stage that exist in The Data Encryption standard (DES) as shown in the following figure:

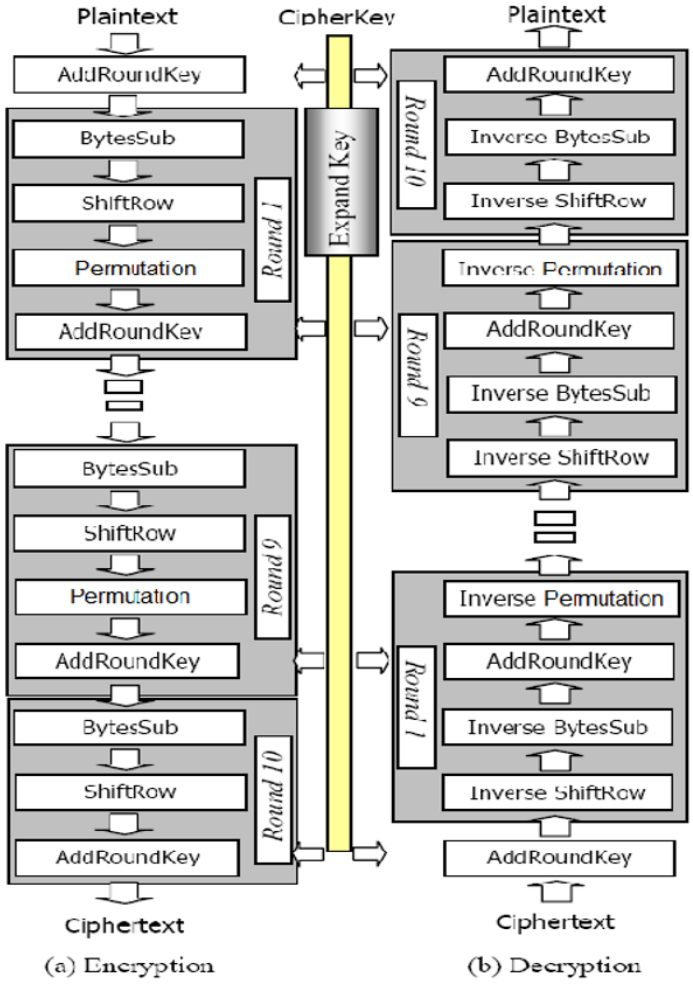

Figure 3-4: the modified scheme (Hameed et al., 2011).

In reference [13] the researchers made an Enhanced Advanced Encryption Standard (E-AES) algorithm, it's propose to achieve more security goals. By adding an extra XOR step which ensure improvements in the encryption process in term of avalanche effect. Where avalanche effect determine how much the output will change if one bit in the input change. So a good avalanche effect when a single bit is changed the hash sum becomes completely different.

Also, they mapped each of the plain text and the key to binary code before encryption process. To make them more secure and complicated. As shown in the following figure.

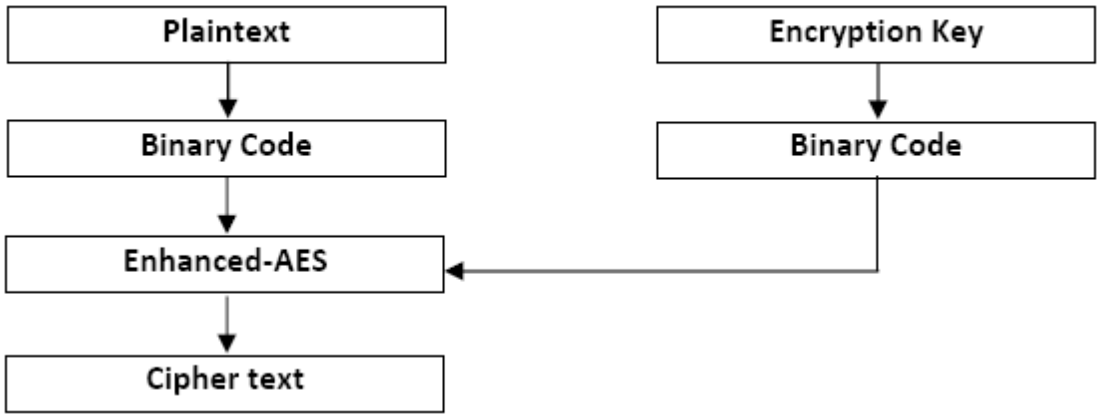

Figure 3-5: Block Diagram of proposed Algorithm.

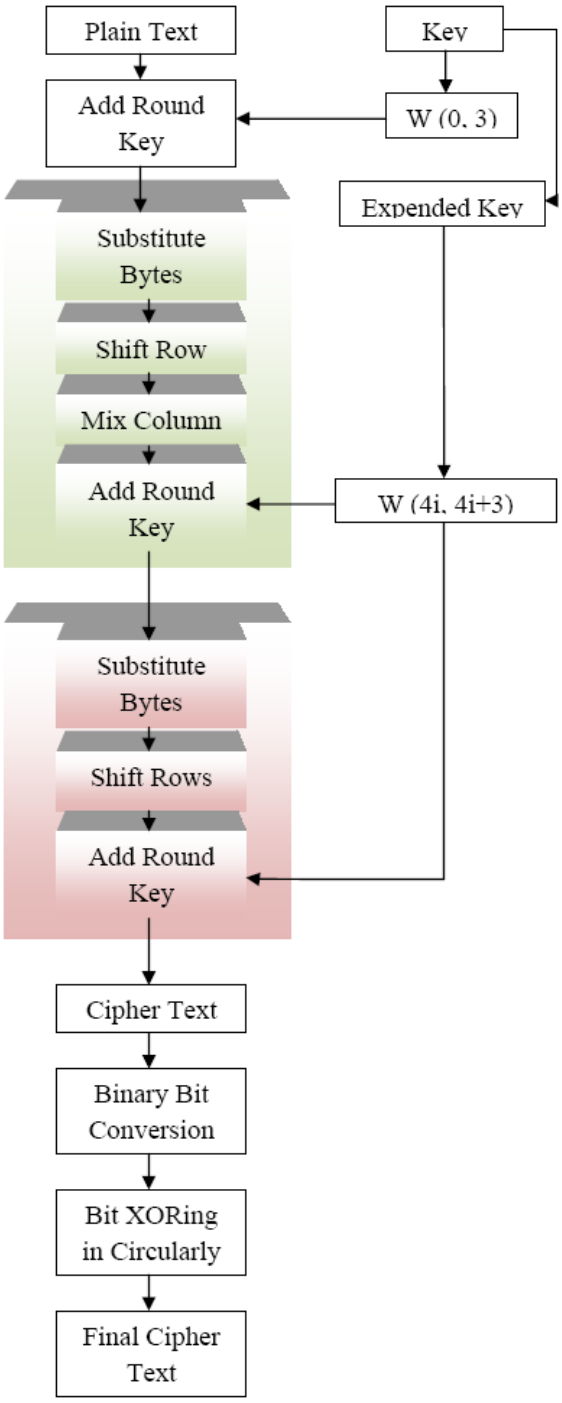

Figure 3-6: Modify AES Architecture.

4 The implementation of Proposed Mechanism for Improving AES Algorithm

4.1 Introduction

This chapter introduces the methodology of the customized encryption approach and its simulation using MATLAB software, as well as the original algorithm simulation.

MATLAB is a high-level programming language with a user-friendly environment for calculations, visualization, and programming.

We can use this platform to design models, algorithms, and apps. Using the platform's built-in arithmetic capabilities, we can obtain the same result faster than spreadsheets or traditional programming languages like C/C++ or Java. [14]

We can build executable code for any computer system in a variety of situations with the MATLAB compiler.

Signal processing and communications, image and video processing, control, test and measurement systems, computational finance, and computational biology are just a few of the applications for this platform.

And that can be done by designing a program to encrypt text and images and decrypt them using

Both the traditional AES encryption algorithm and the improved AES encryption algorithm, this mechanism is simulated using the MATLAB environment and using a computer with the following specifications:

(CPU: Intel Core i7, System: Windows 7 64-bit, RAM: 8GB). [15]

4.2 MATLAB simulation of the original AES algorithm

4.2.1 Encryption simulation steps:

4.2.1.1 The Sub-Bytes Step:

In this step we create a function called **"SubBytes"** that take one parameter "state" matrix and return it after substitution of bytes with S-Box bytes.
The following figure show the implementation of this step.

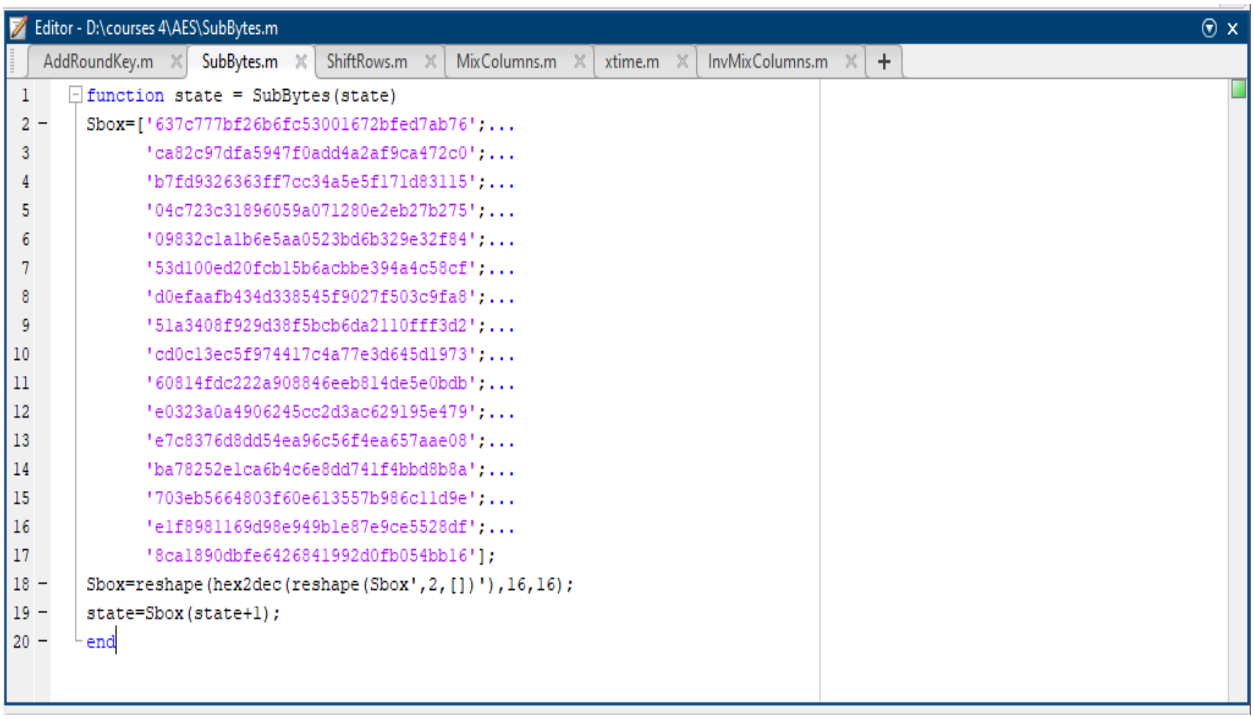

Figure 4-1: Sub-Bytes implementation.

4.2.1.2 The Shift-Rows Step:

In this step we create a function called **"ShiftRows"** that take two parameter "state" matrix and "w" and return the state matrix after madding the needed shifting for each line of matrix by the circular shifting is to the left.

The following figure show the implementation of this step.

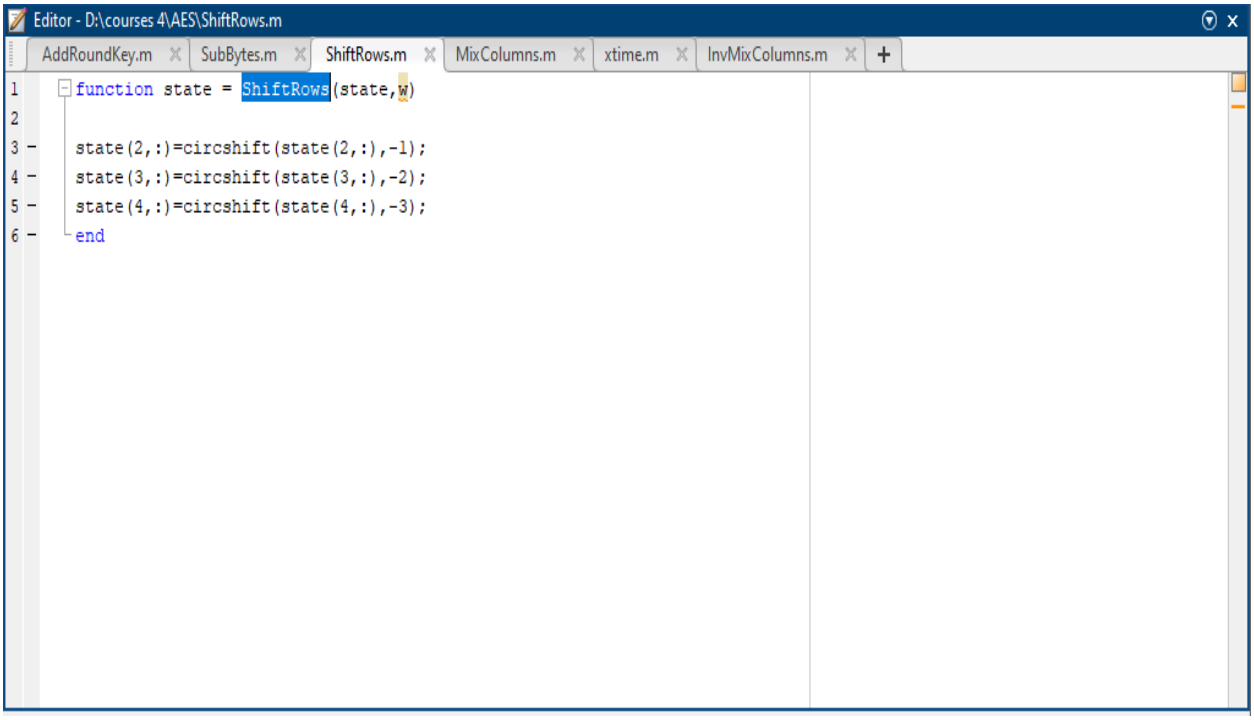

Figure 4-2: Shift-Rows implementation.

4.2.1.3 The Mix-Columns Step:

In this step we create a function called **"MixColumns"** that take one parameter "state" matrix and return the state matrix after multiplying each column of the resulting state matrix by a square matrix dedicated for encryption process with the same dimensions of the state matrix.

The following figure show the implementation of this step.

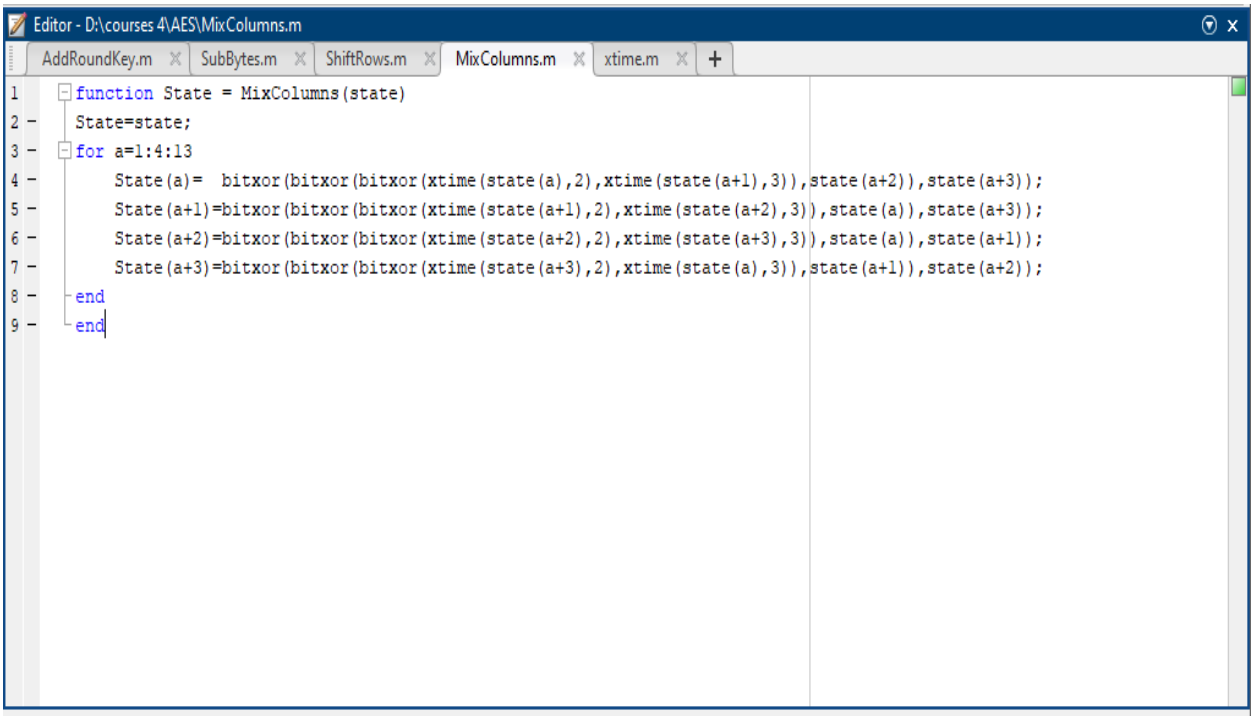

Figure 4-3: The Mix-Columns implementation.

4.2.1.4 Add round key Step:

In this step we create a function called "**AddRoundKey**" that take two parameters "state" matrix and the key of the current round. It returns the state matrix after making binary XOR on each four bytes of state matrix with one WORD of keys expansion matrix.

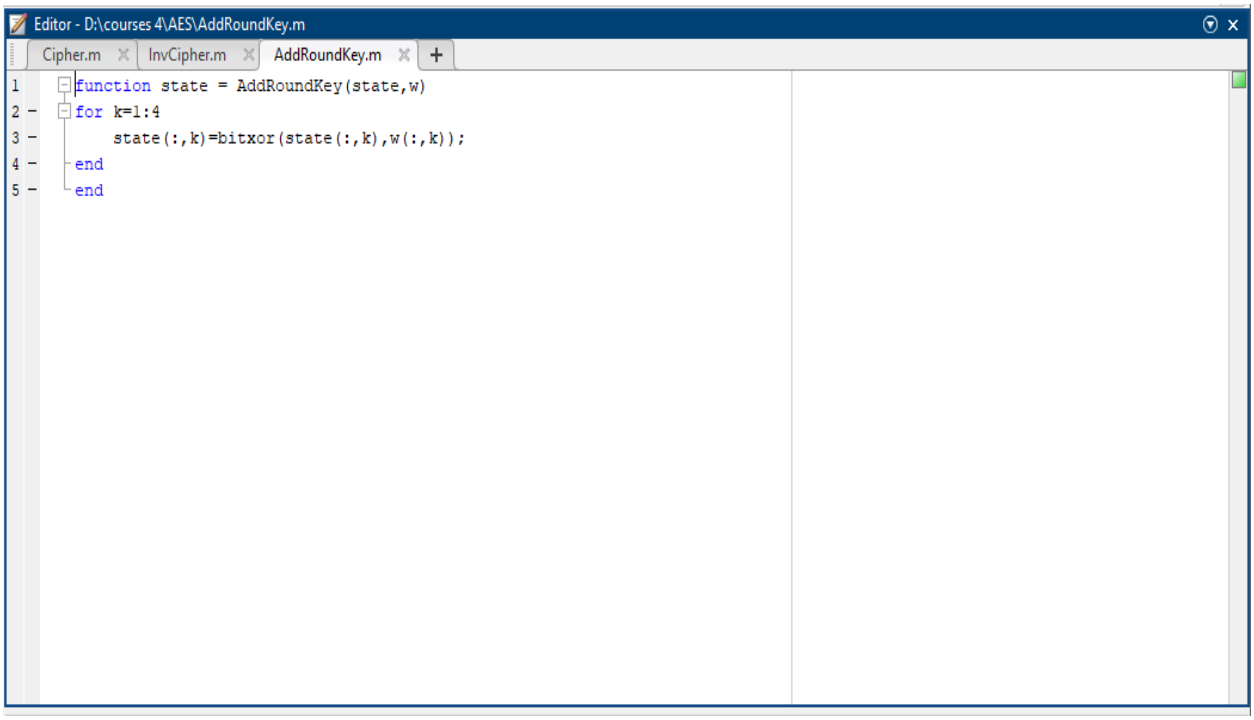

Figure 4-4 : The Add round key implementation.

4.2.1.5 Cipher Step:

In this step we create a function called “Cipher” that take two parameters: the original key “key” and the input of the plain text “In”. In this function we call all the previous encryption functions that are essential for the encryption process, that are SubBytes, ShiftRows, MixColumns, and AddRoundKey. That will be called 10 times in case 128 bit key. In addition to calling AddRoundKey function in the initial stage. This function return the cipher text.

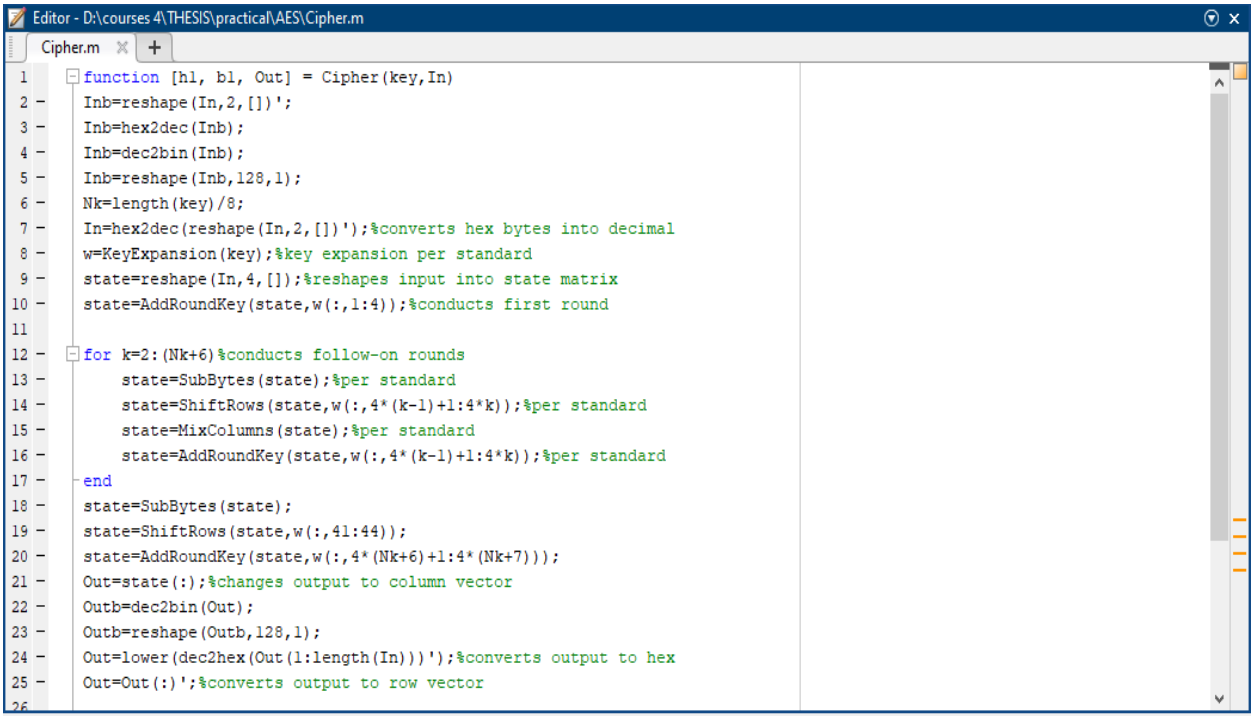

Figure 4-5 : The Cipher implementation.

4.2.2 Decryption simulation steps:

4.2.2.1 The Inverse Sub-Bytes Step:

In this step we create a function called "**InvSubBytes**" that take one parameter "state" matrix and return it after substitution of bytes with inverse S-Box bytes.

The following figure show the implementation of this step.

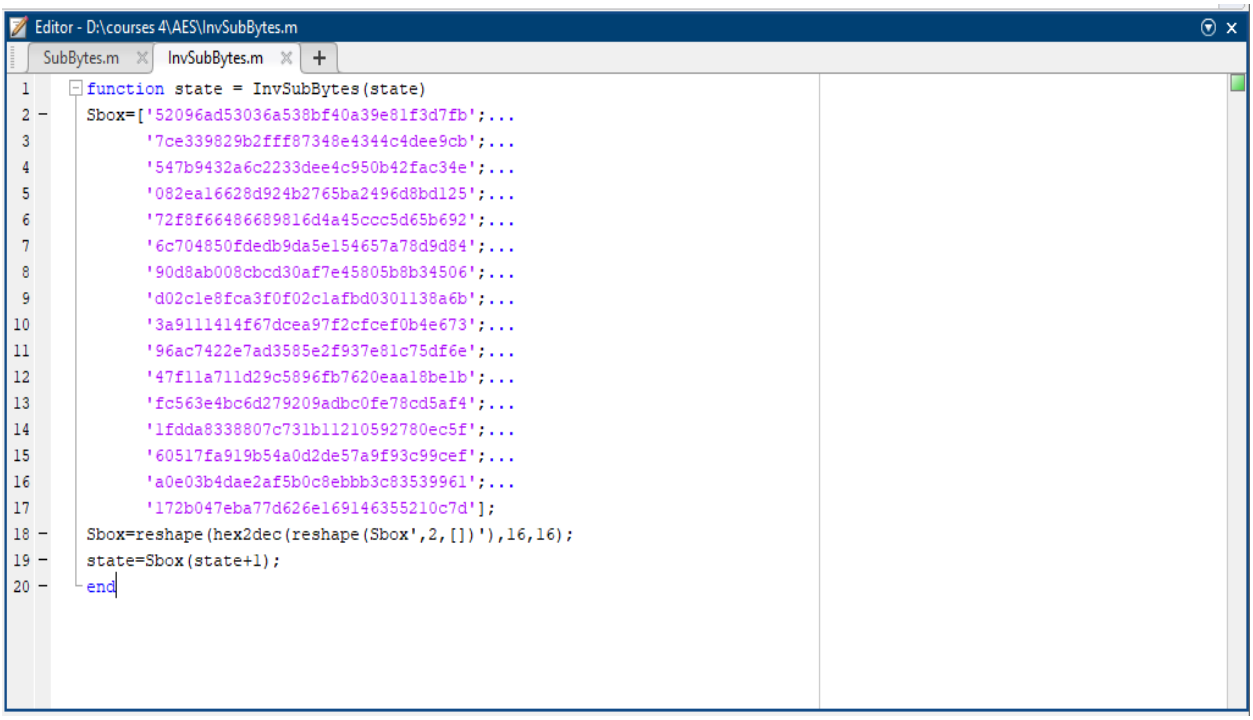

Figure 4-6 : Inverse Sub-Bytes implementation.

4.2.2.2 The Inverse Shift-Rows Step:

In this step we create a function called "**InvShiftRows**" that take two parameter "state" matrix and return the state matrix after madding the needed shifting for each line of matrix by the circular shifting is to the right.

The following figure show the implementation of this step.

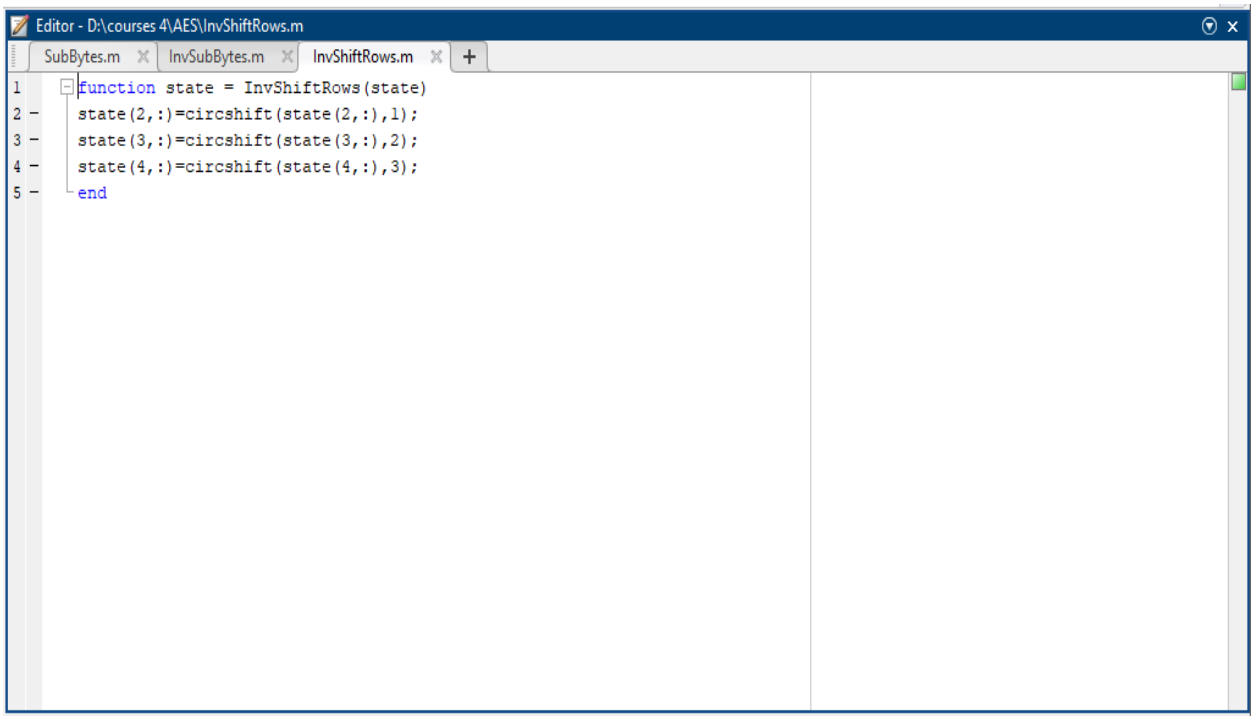

Figure 4-7 : Inverse Shift-Rows implementation.

4.2.2.3 The Inverse Mix-Columns Step:

In this step we create a function called **"InvMixColumns"** that take one parameter "state" matrix and return the state matrix after multiplying each column of the resulting state matrix by a square matrix dedicated for decryption process with the same dimensions of the state matrix.

In this step we need another function called "xtime" that take two parameter "x" and "c".

The following figure show the implementation of this step.

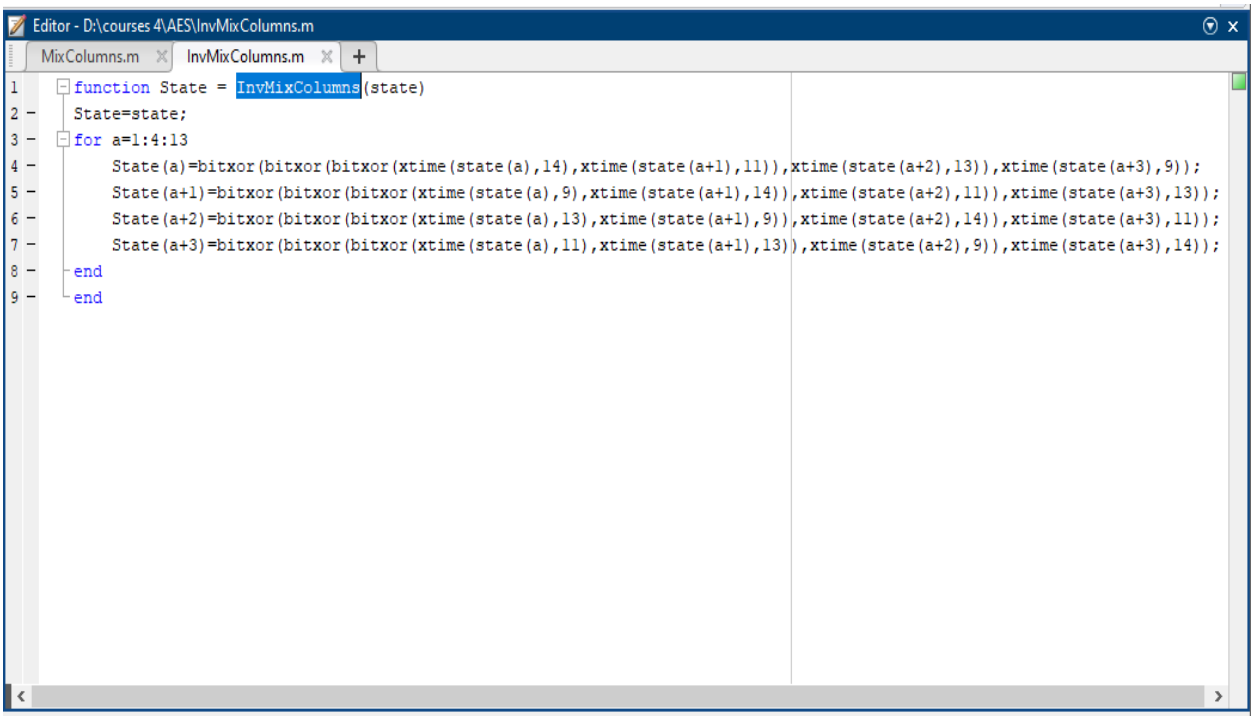

Figure 4-8 : Inverse Mix-Columns implementation.

4.2.2.4 Add round key Step:

In this step we use the same function **"AddRoundKey"** that created in the encryption process. So the same code will be applied to the decryption process without any change.

4.2.2.5 Inverse Cipher Step:

In this step we create a function called **"InvCipher"** that take two parameters: the original key "key" and the input of the cipher text "In". In this function we call all the previous inverse functions that are essential for the decryption process, that are InvSubBytes, InvShiftRows, InvMixColumns, and AddRoundKey. That will be called 10 times in case 128 bit key. In addition to calling AddRoundKey function in the initial stage. This function return the plain text.

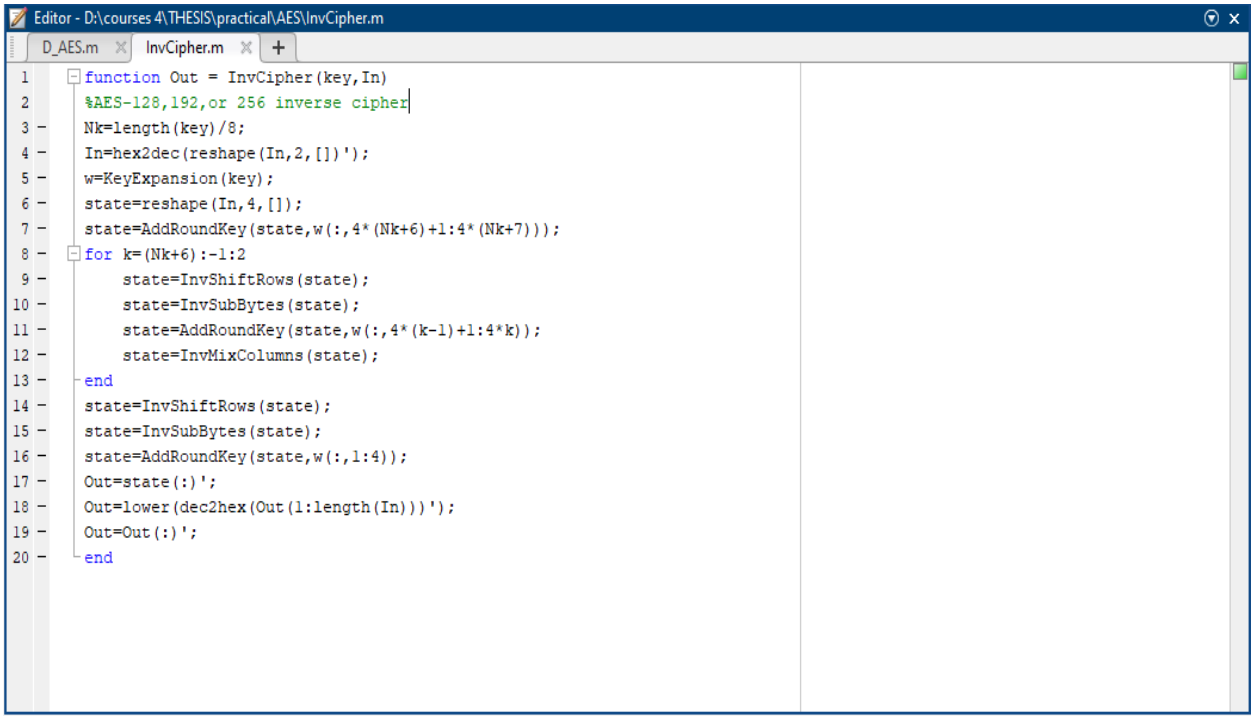

Figure 4-9 : The Inverse Cipher implementation.

4.2.3 Key Expansion simulation steps:

In this step we create a function called "**key_expansion**" that take one parameters: the original key "key" and return "w" linear array of 44 words (11 keys).

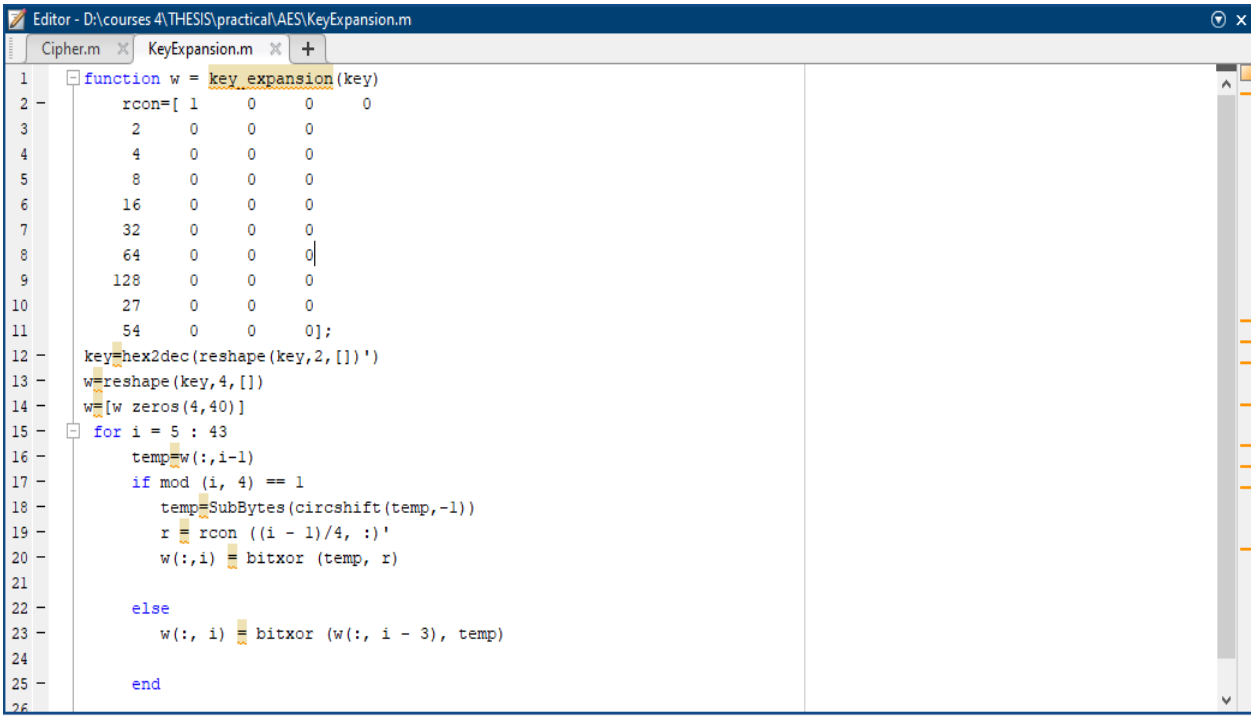

Figure 4-10 : The Key Expansion implementation.

4.2.4 Constructing the graphical user interface (GUI):

A graphical user interface (GUI) is a visual interface that allows a user to interact with an application without having to know the language.

This is accomplished by giving simple controls. The controls are the buttons that the user presses in order to get a specific result. A graphical user interface (GUI) is an event-driven application. This is due to the fact that it accepts input at any moment and uses callback functions to execute the program and return results.

MATLAB offers a nice tool to create and design a graphical user interface, we can access this tool by typing guide in the workspace. Then a new window opens up. We can add and design the needed components in this window. The following figure show our system graphical design. [16]

Figure 4-11 : constructing the graphical user interface

4.2.5 Writing the callback function of the components:

When the designed GUI is running, the .m file is automatically generated before the .fig file shows up. When the user click a one button it will execute the call back function, and show the related result of clicking this button. [16]

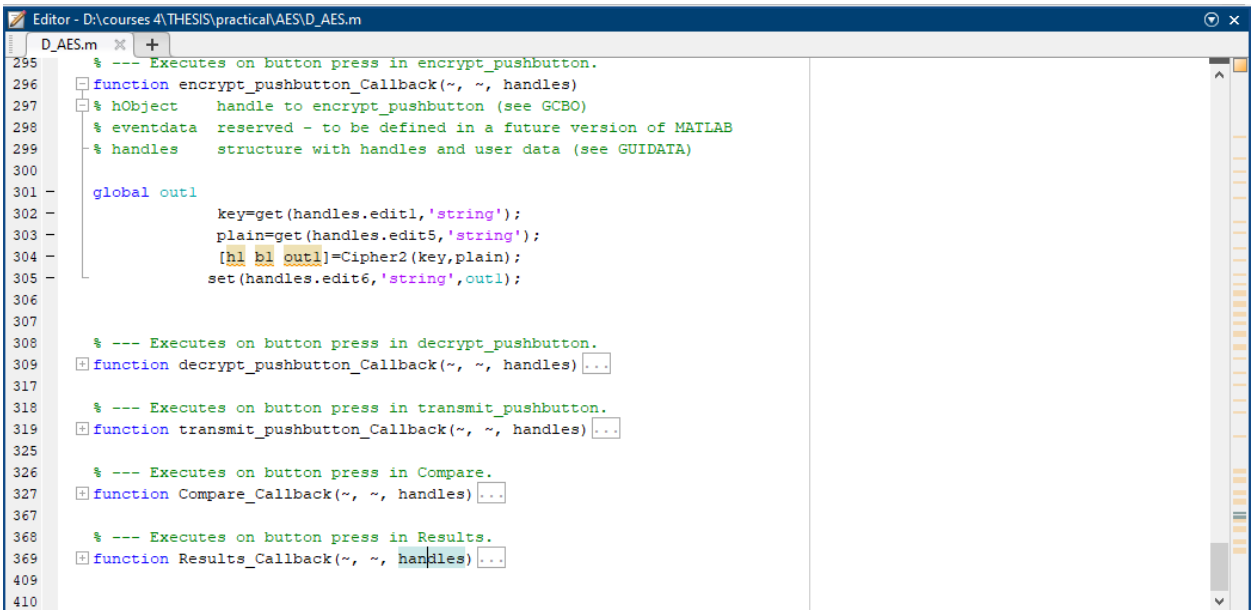

Figure 4-12 : writing the callback function of the components

Finally we need to check our code for the original AES work probably, that means we get the same plain text in both transmitter and receiver. We will apply 32 Fs as a standard input, we expect 32Fs as output.

The following figure show that.

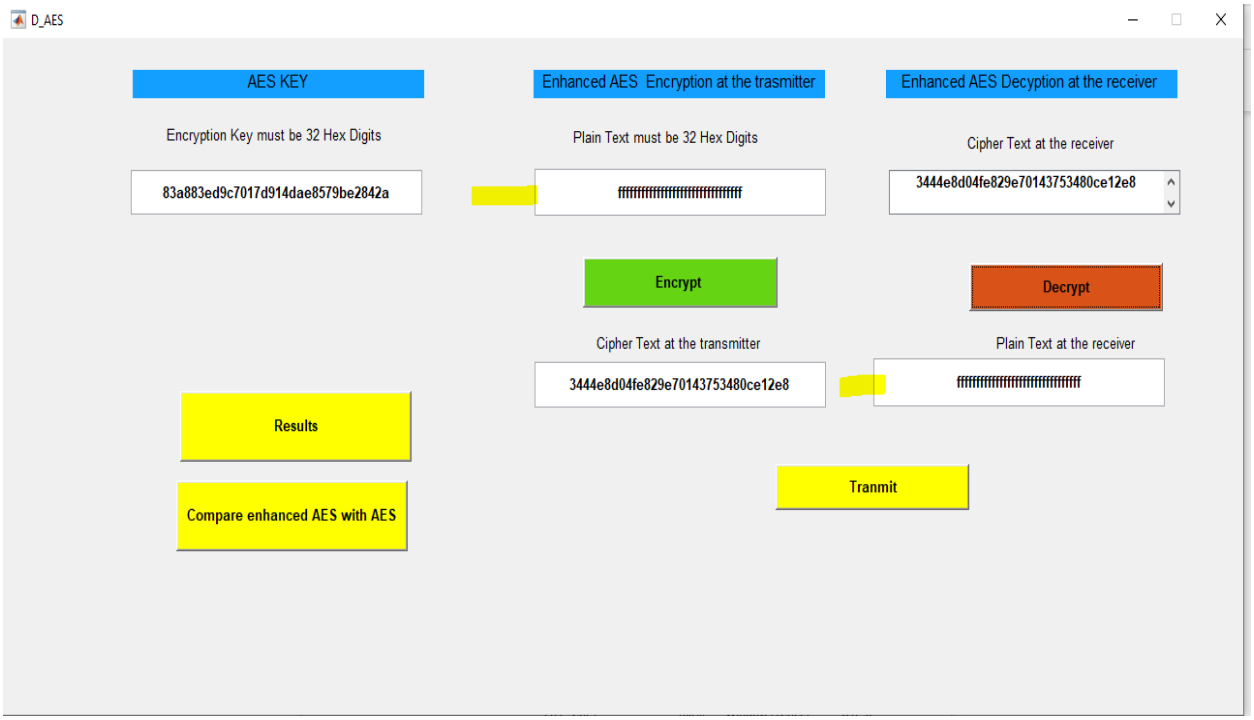

Figure 4-13 : same plain text in both Transmitter and Receiver with Cipher /InCipher

4.3 MATLAB simulation of the D AES algorithm

4.3.1 Enhancing Key Expansion algorithm:

In this step we make enhancement for the original key Expansion algorithm. First we create matrix for each round, each matrix contains two elements, the first element is round constant key and the second element is one of the key words. After creating this matrix we make a binary XOR between each round matrix and the previous output of this process.

For the first round the previous output will be a matrix of zeros.

That leads to more complexity in key expansion algorithm. The key words was chosen in a specific order for each round after make many experiments and by comparison the results the suggested order was the best one in term of security.

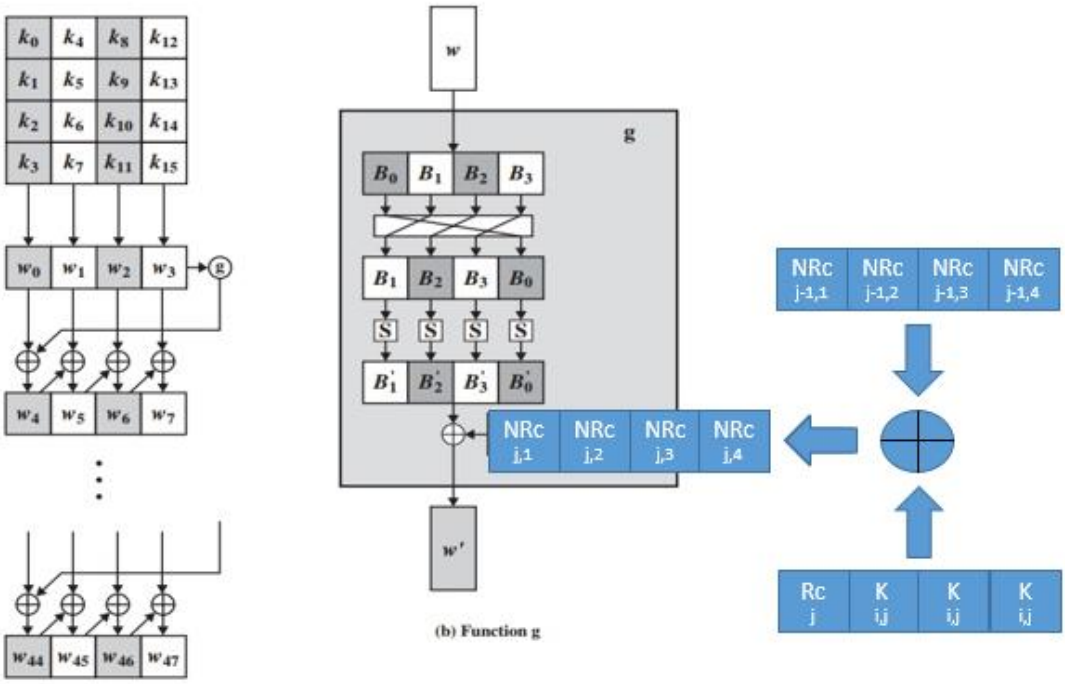

Figure 4-14 : Proposed Enhancing Key Expansion algorithm

The following figure show the implementation of enhancing key expansion algorithm.

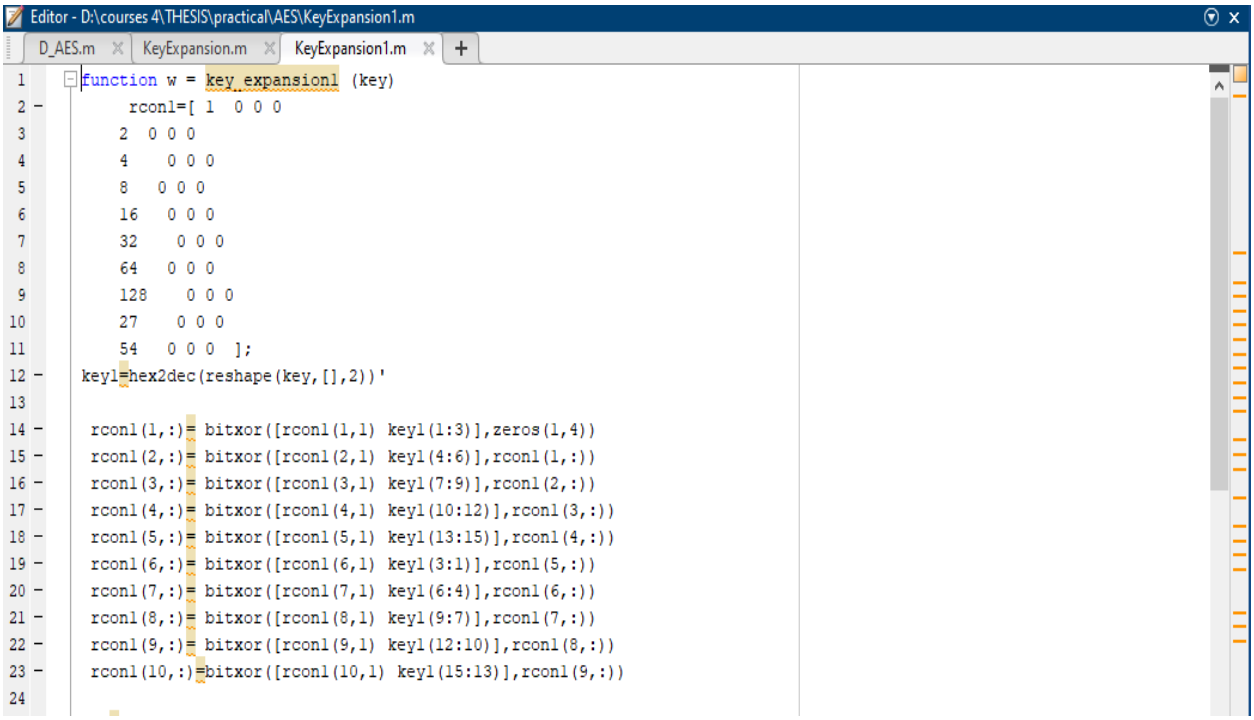

Figure 4-15 : Enhancing Key Expansion algorithm implementation

After applying the suggested algorithm we have to modify the cipher algorithm in the original AES code to take Enhancement Key Expansion parameter instead of the original one.

The following figure show the modification.

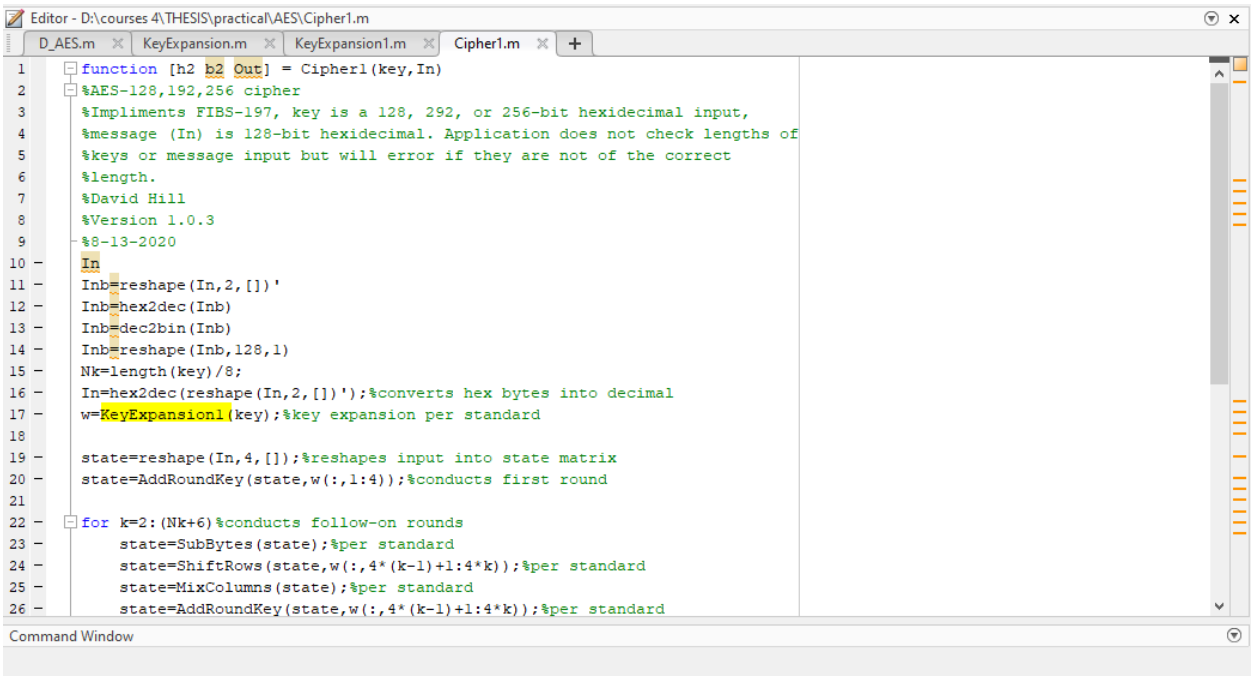

Figure 4-16 : Modified Cipher implementation (Cipher1)

Also we have to modify the inverse cipher algorithm to have the same plain text after decryption process. The modification will simply involve the Enhanced Key Expansion instead of the original one.

The following figure show the modification.

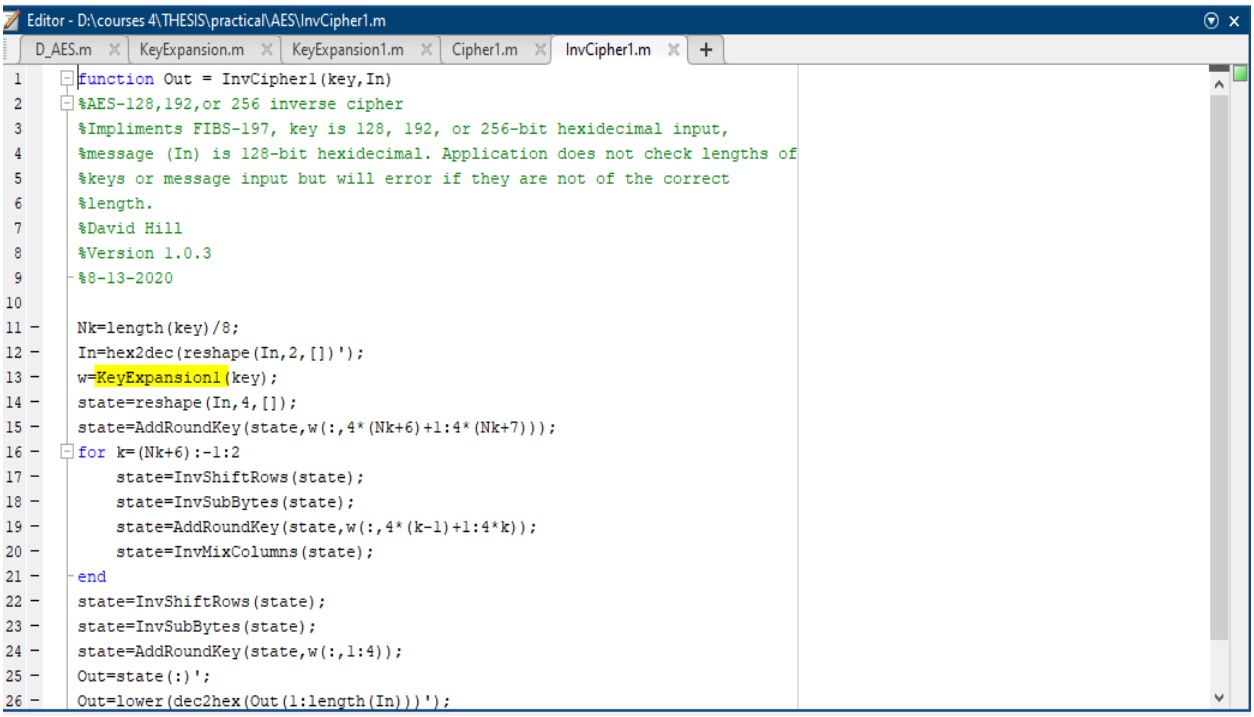

Figure 4-17 : Modified InvCipher implementation (InvCipher1)

The last thing we have to check our algorithm work properly with the graphical user interface GUI and callback functions, so we modify the encrypt and decrypt call back function in the main code D_AES.m to check if the same plain text produced after decryption.

The following figure show the modification:

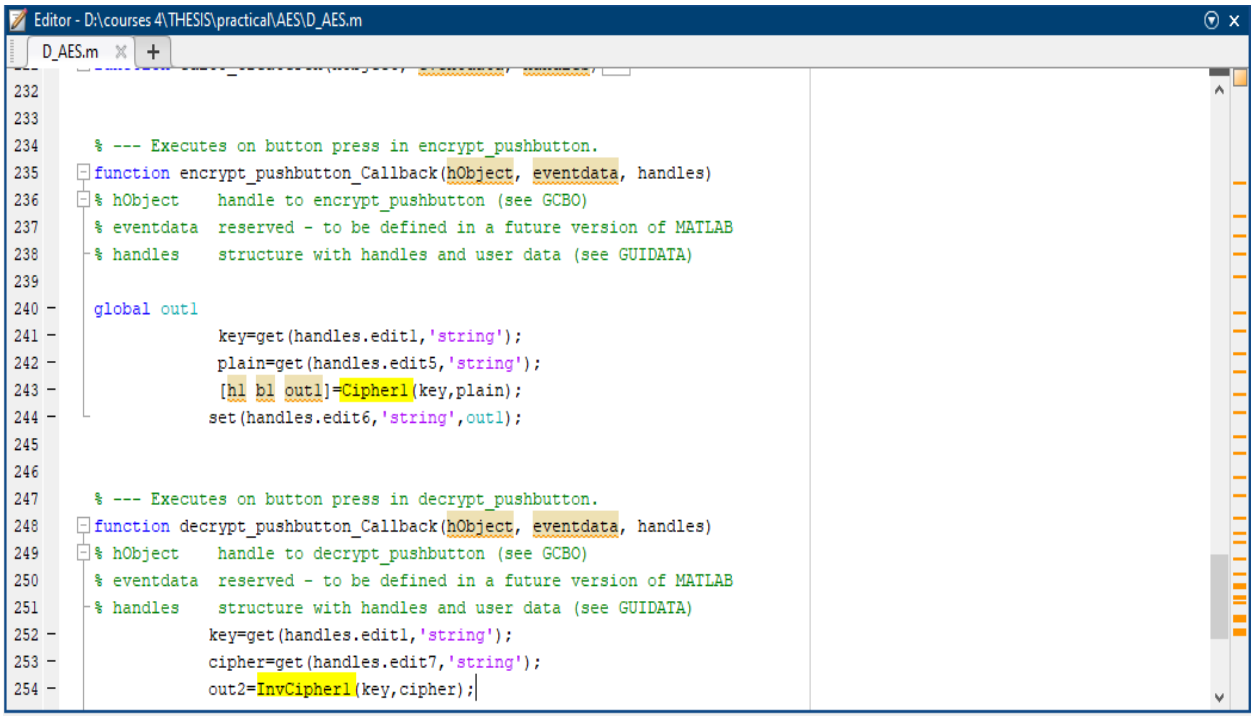

Figure 4-18 : Modified Main code D_AES.m (Cipher1/InvCipher1)

The following figure show the plain text in both the transmitter and receiver is the same on the graphical user interface GUI after applying standard input.

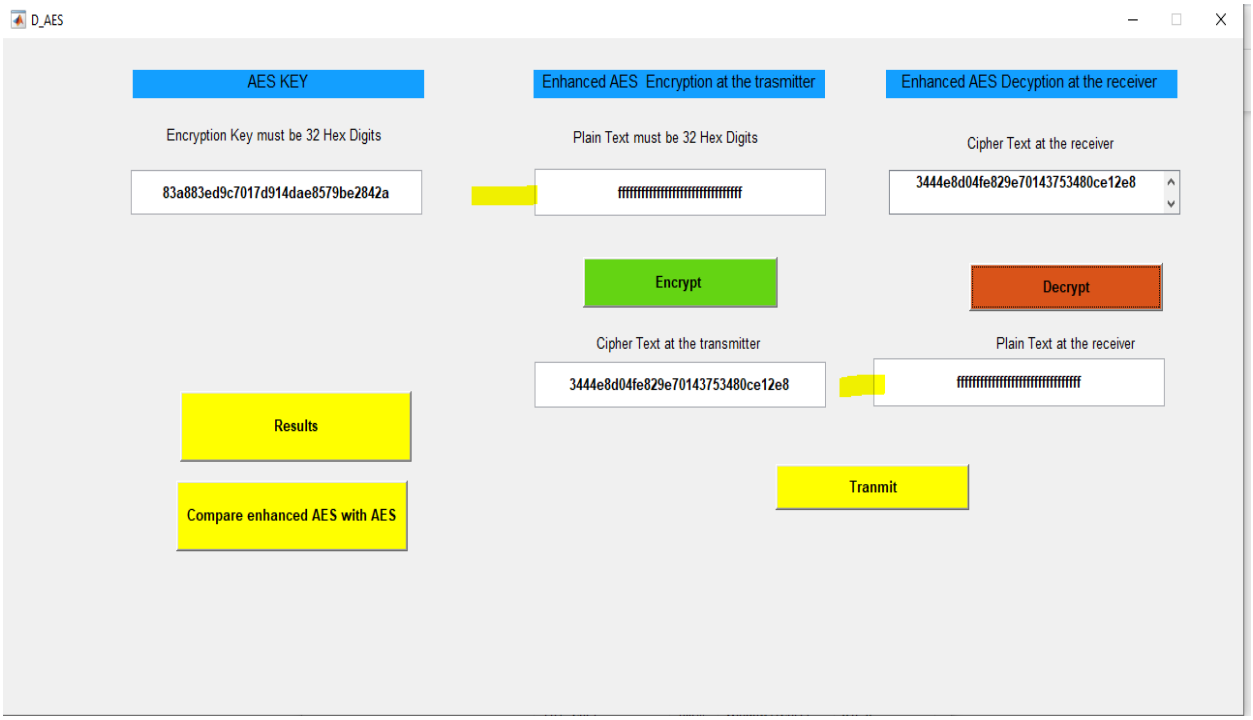

Figure 4-19 : same plain text in both Transmitter and Receiver with Cipher1 /InCipher1

4.3.2 Enhancing Shift-Rows algorithm:

In this step we add another enhancement for the previous step by modifying the original Shift-Rows algorithm. The original Shift Rows algorithm make circular shifting to the left by zero, one, two or three places depending on state matrix line number. The new modification introduce circular shifting to the left by word index of the key instead of fix shifting number. The suggested distributions of key words for shifting will be as follow:

-The first line of state matrix will be shifted to the lift by: 1:4 of the key. [First word]
-The second line of state matrix will be shifted to the lift by: 5:8 of the key. [Second word]
-The third line of state matrix will be shifted to the lift by: 9:12 of the key. [Third word]
-The fourth line of state matrix will be shifted to the lift by: 13:16 of the key. [Forth word]

That will leads to more complexity in each round of the encryption process. So in this case the number of places shifting will not be predictable as long as it depends on the round key.

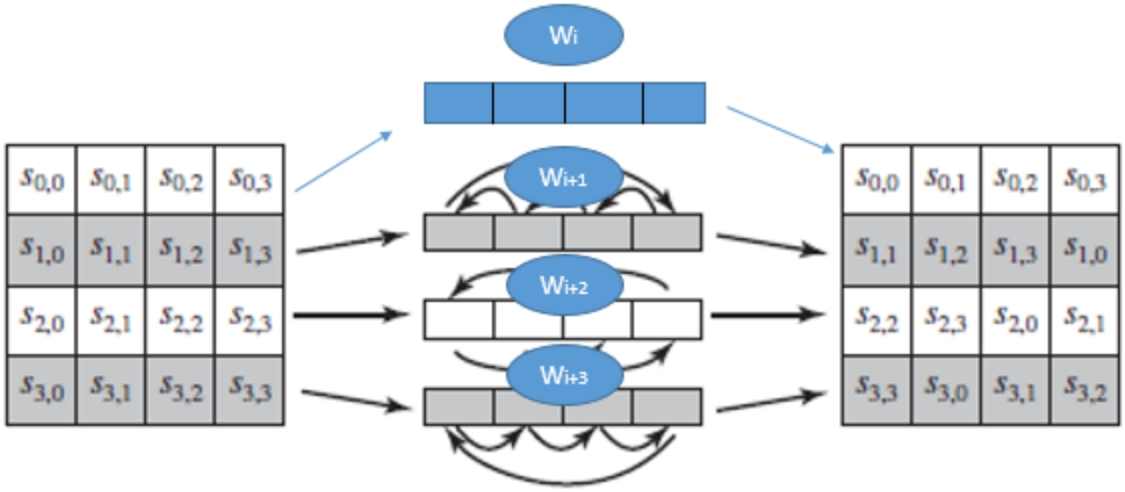

Figure 4-20 : Proposed Enhancing Shift-Rows algorithm (Shift-Rows2)

The following figure show the implementation of enhancing Shift-Rows algorithm.

Editor - D:\courses 4\THESIS\practical\AES\ShiftRows2.m

ShiftRows2.m | ShiftRows.m | D_AES.m

```
function state = ShiftRows2(state,w)
stateb=state

z1=dec2bin(w)
z11=z1(1:4,:)
z22=z1(5:8,:)
z33=z1(9:12,:)
z44=z1(13:16,:)
z111=reshape(z11,32,[]);
z222=reshape(z22,32,[]);
z333=reshape(z33,32,[]);
z444=reshape(z44,32,[]);
c1=0;
c2=0;
c3=0;
c4=0;
for p=1:32
        if z111(p)=='1'
            c1=c1+1;
        end

end
for p=1:32
```

Figure 4-21 : Enhancing Shift-Rows algorithm (Shift-Rows2) implementation

Same process applied in Inverse Shift-Rows algorithm, to get the same plain text in the receiver side.

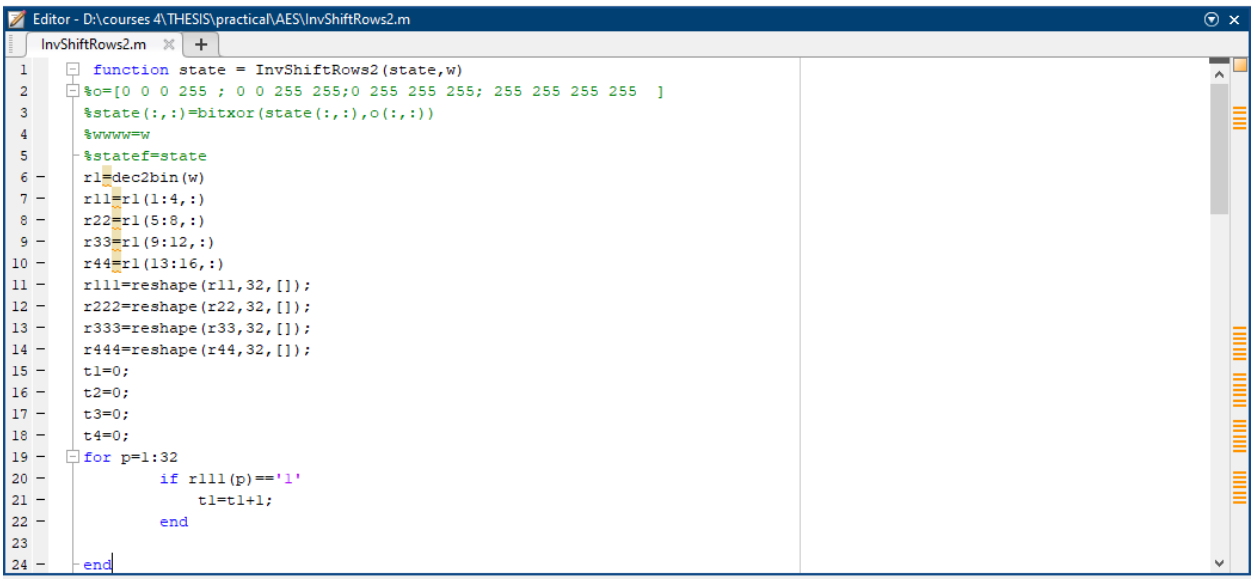

Figure 4-22 : Enhancing Inverse Shift-Rows algorithm (InvShiftRows2)

After applying the suggested algorithm we have to modify the last modified cipher algorithm Cipher1 to take Enhancing Shift-Rows parameter instead of the original one.

The following figure show the modification.

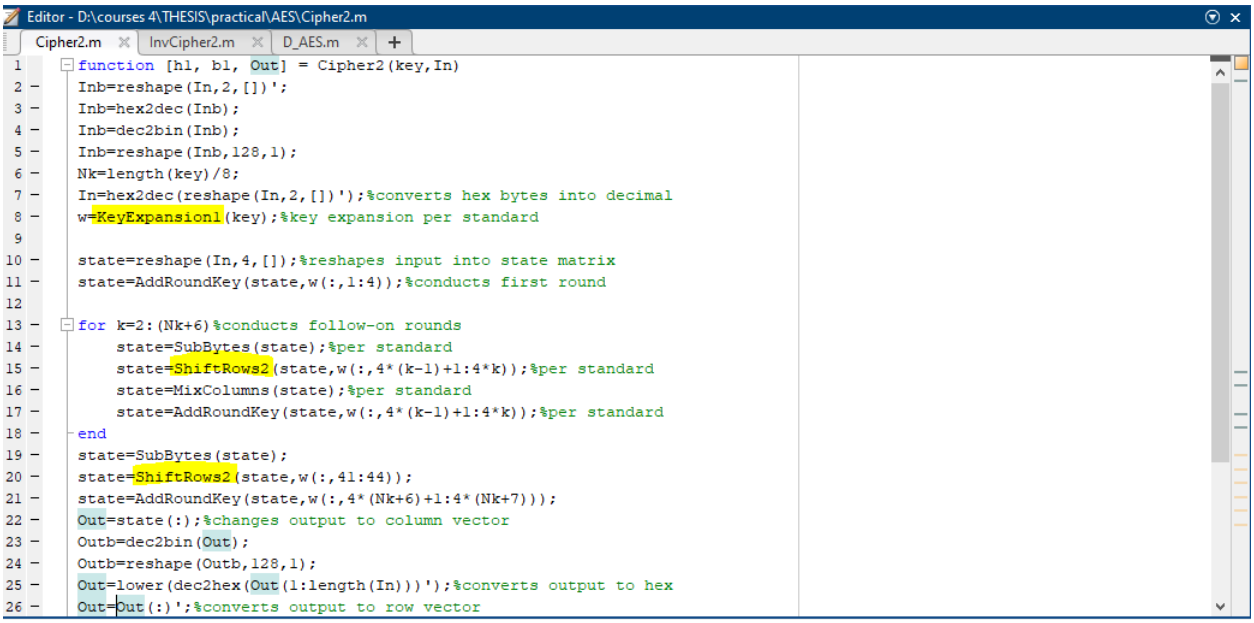

Figure 4-23 : Modified Cipher implementation (Cipher2)

Also we have to modify the inverse cipher algorithm to have the same plain text after decryption process. The modification will simply involve the Enhanced Shift-Rows instead of the original one. The following figure show the modification.

Editor - D:\courses 4\THESIS\practical\AES\InvCipher2.m

InvCipher2.m

```
function Out = InvCipher2(key,In)
%AES-128,192,or 256 inverse cipher
%Impliments FIBS-197, key is 128, 192, or 256-bit hexidecimal input,
%message (In) is 128-bit hexidecimal. Application does not check lengths of
%keys or message input but will error if they are not of the correct
%length.
Nk=length(key)/8;
In=hex2dec(reshape(In,2,[])');
w=KeyExpansion1(key);
state=reshape(In,4,[]);
state=AddRoundKey(state,w(:,4*(Nk+6)+1:4*(Nk+7)));
for k=(Nk+6):-1:2
    state=InvShiftRows2(state,w(:,4l:44));
    state=InvSubBytes(state);
    state=AddRoundKey(state,w(:,4*(k-1)+1:4*k));
    state=InvMixColumns(state);
end
state=InvShiftRows2(state,w(:,4l:44));
state=InvSubBytes(state);
state=AddRoundKey(state,w(:,1:4));
Out=state(:)';
Out=lower(dec2hex(Out(1:length(In)))');
Out=Out(:)';
end
```

Figure 4-24 : Modified InvCipher implementation (InvCipher2)

As previous modification of Key Expansion we also have to modify the main code D_AES.m to include the new Cipher and InCipher functions.

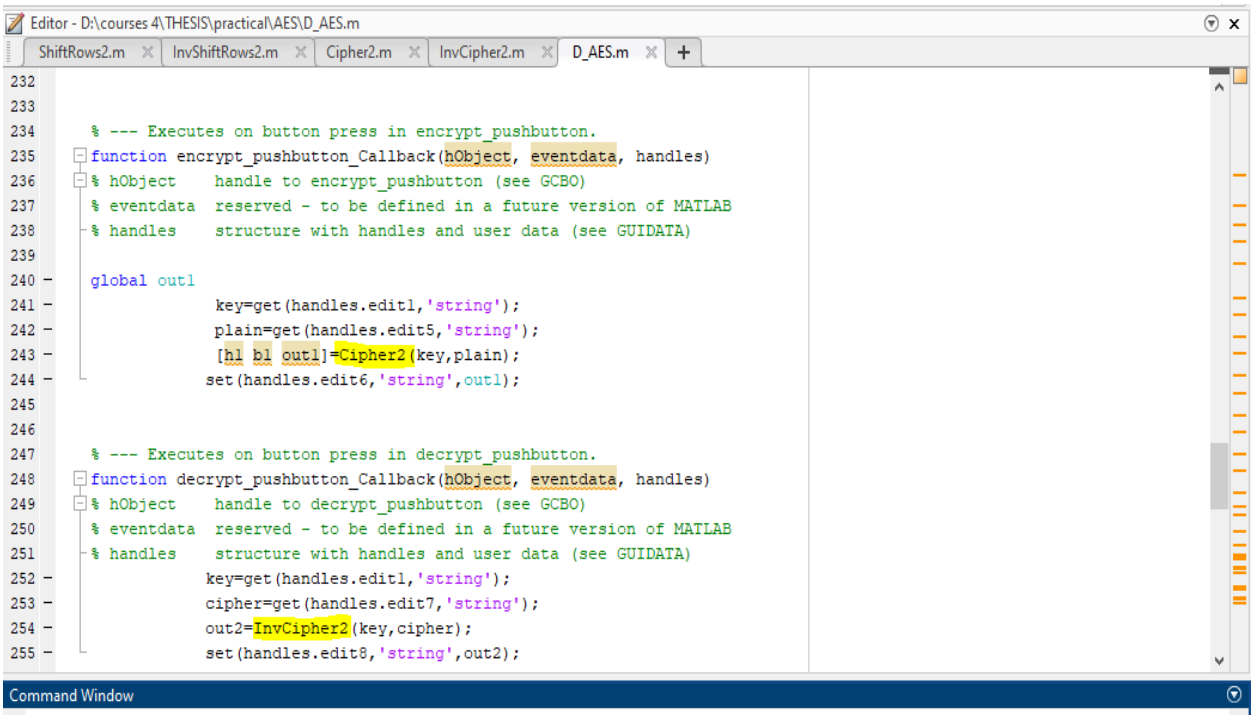

Figure 4-25 : Modified Main code D_AES.m (Ciphe2/InvCipher2)

The following figure show the plain text in both the transmitter and receiver is the same on the graphical user interface GUI after applying standard input.

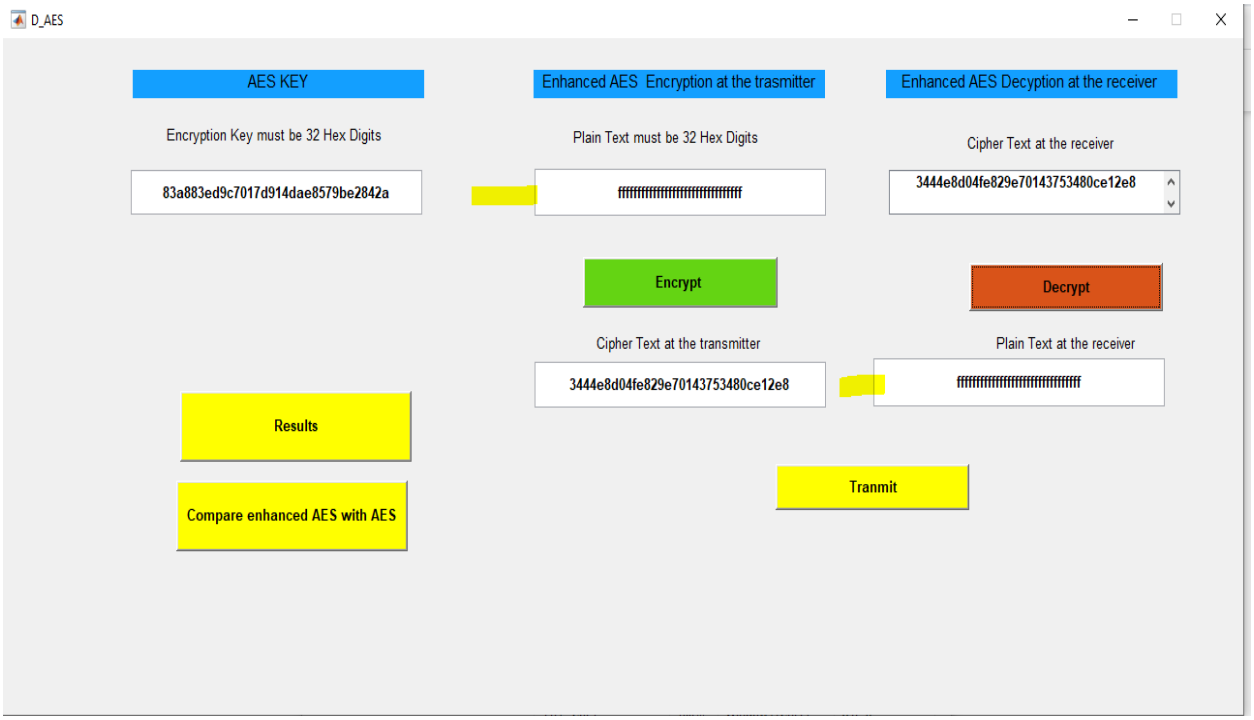

Figure 4-26 : same plain text in both Transmitter and Receiver with Cipher2 /InCipher2

5 The Results of implementation the enhanced D AES Algorithm

In this chapter the performance of the enhanced algorithm will be evaluated by studying some essential parameters and make a comparison between the original AES and the enhanced DAES.

5.1 Hamming distance parameter

Hamming distance parameter calculate the number positions of corresponding symbols which are different between two strings of equal length. [17]

The following figure demonstrate this parameter.

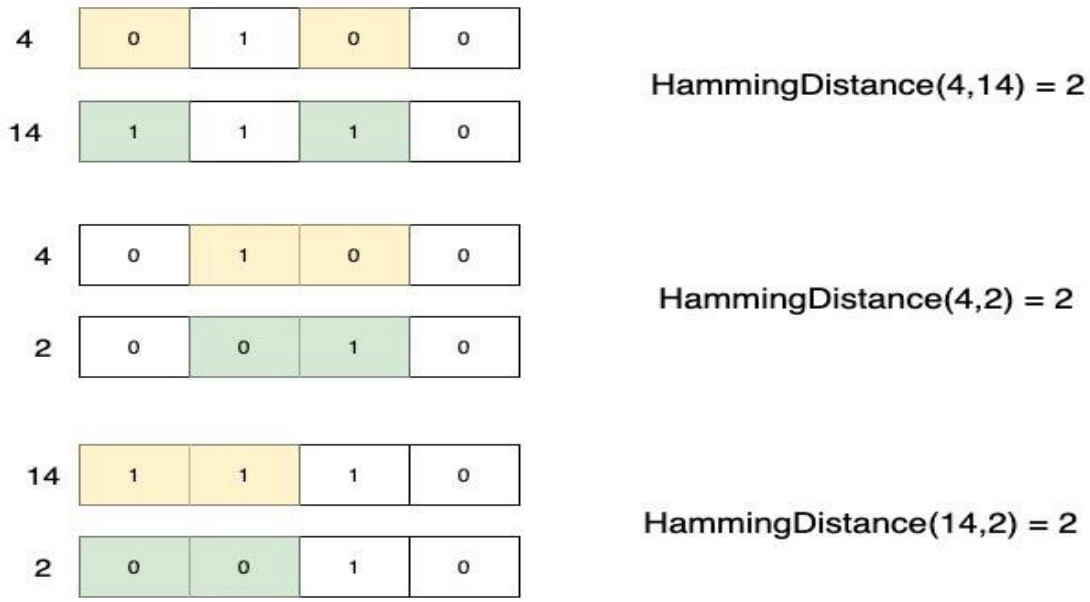

Figure 5-1 : Hamming distance parameter.

So in cryptography the hamming distance is the number of different the number positions of corresponding symbols which are different between the plain text and the encrypted text. And the encryption algorithms with high numbers means are more secure than those with lower ones. By applying a sequences of plain texts with constant key (32 Fs). The result will be as follows:

ID	The plain texts (32 Hex Digits)	Hamming distance in the original AES	Hamming distance in the D-AES Phase 1	Hamming distance in the D-AES Phase 2
1	a95c145dff3ff291307e8582adf0f52f	68	68	70
2	9e90be643882e228c52ce177de8c7a41	72	72	74
3	8dc614d4de94282fdce6d494f3d83a60	62	74	75
4	76eb0cfa51a91dd0603fec8257f0151a	67	67	65
5	5bc37ae4668e0f4ec0648ca87f4ebb05	77	70	72
6	35200f4237a2890b42069ea9b80802d3	63	64	65
7	637ba4a6d25ea088a18d1122bb793e2f	67	64	66
8	6214a2e936da8e2117e25484f8991f4e	65	73	77
9	2437420fad20980906a930519b559834	64	69	75
10	05d2ec4e13149bfa6920fab4419751d4	73	72	74
AVG		**68**	**69**	**71**

Table 5-1: a sequences of plain texts with constant key- Hamming distance

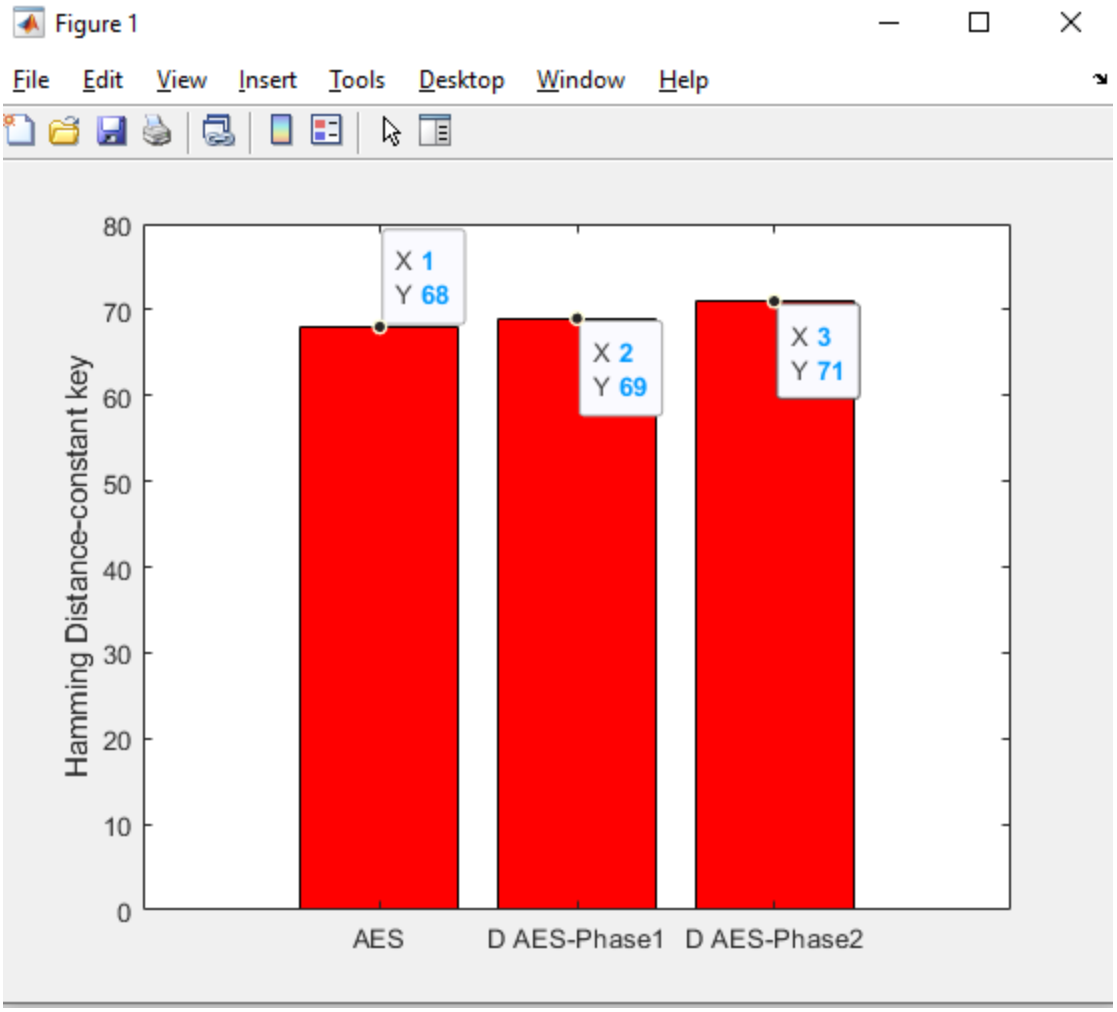

Figure 5-2 : Average Hamming distance parameter-constant key

-Enhanced Percentage: 1.5 % in DAES-phase1- ***constant key.***

- Enhanced Percentage: 4.3 % in DAES-phase2- ***constant key.***

-By applying a sequences of keys with constant plain text (32 Fs). The result will be as follows:

ID	The key (32 Hex Digits)	Hamming distance in the original AES	Hamming distance in the D-AES Phase 1	Hamming distance in the D-AES Phase 2
1	2646e60a2ba7dcd2508a6eac28b73010	68	68	70
2	4f04d346dfe774d7e4d80b353eafffc6	75	72	74
3	ff908bc64d4378c8a7b05a8dce7e3dcc	61	75	78
4	f7617a7c96e974dea278f8b40e896ae9	60	67	67
5	8c410243919e54c308b320da151725e5	77	74	76
6	4f361575d0cd17f70b58b53be8b7abf4	57	68	69
7	7c0ce8f90da9fdc1880f04ad31307089	60	64	66
8	8366a53e9a615700c6736b432650154f	59	73	78
9	9daad39ce0abdd349a8fba6994a322cb	62	69	75
10	aa7f2bdb7ab42fc44fb9d69fac7286ec	71	78	77
AVG		**65**	**71**	**73**

Table 5-2: a sequences of keys with constant plain text -Hamming distance

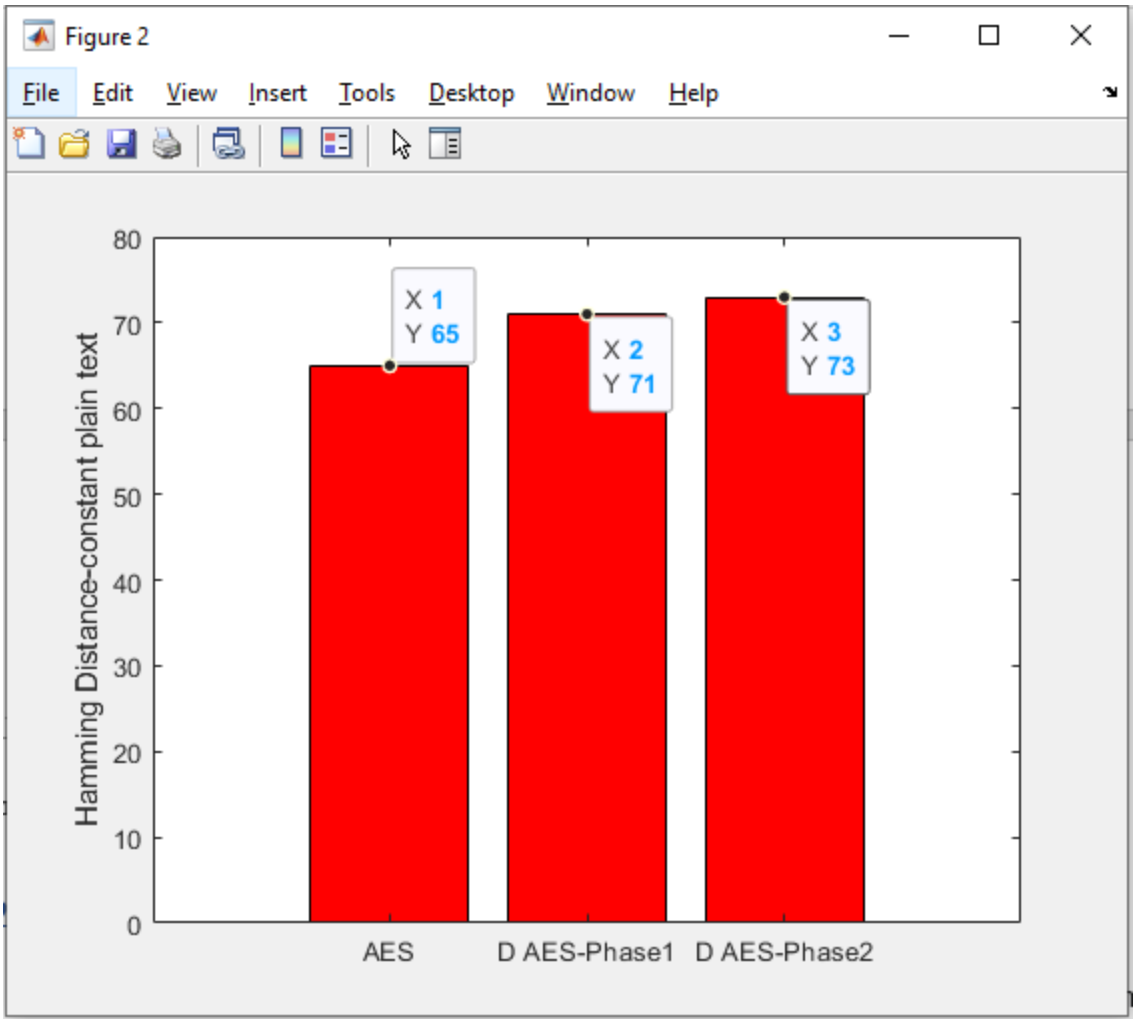

Figure 5-3 : Average Hamming distance parameter-constant plain text

-Enhanced Percentage: 8.8 % in DAES-phase1-*constant plain text.*

- Enhanced Percentage: 11.6 % in DAES-phase2- *constant plain text.*

5.2 Avalanche effect parameter

The avalanche effect parameter is a desirable attribute of cryptographic algorithms in which if an input is slightly changed (e.g., by flipping a single bit), the output changes dramatically (e.g., half the output bits flip). A slight change in either the key or the plaintext should induce a significant change in the cipher text in high-quality block ciphers.[18]

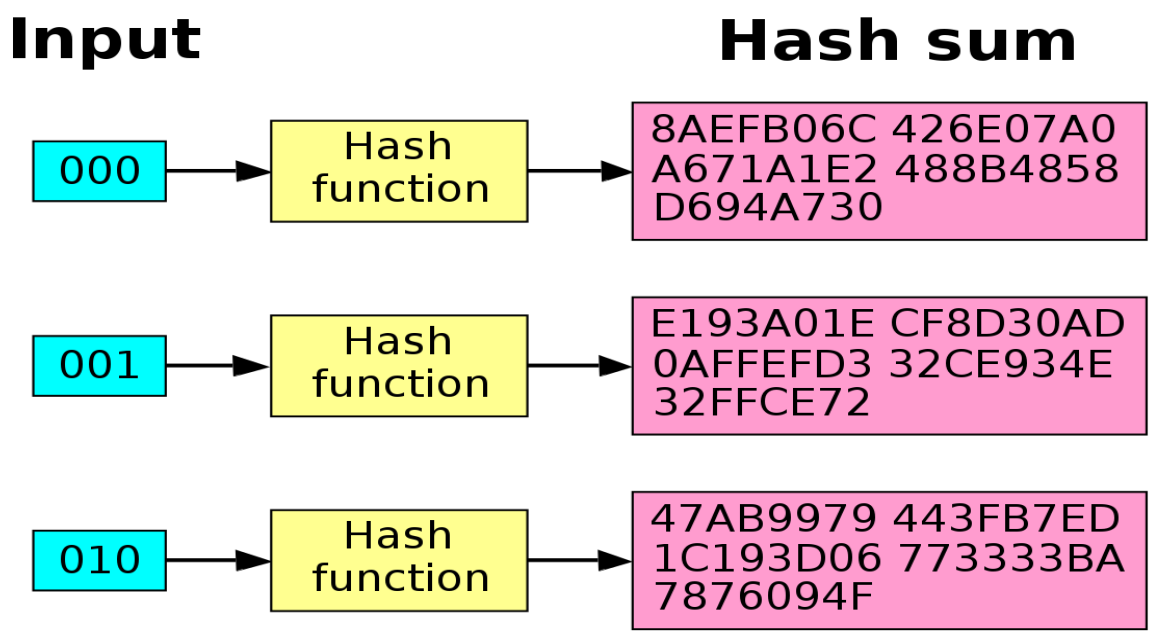

Figure 5-4 : Avalanche effect parameter.

Avalanche Effect= (Number of Changed bit in ciphertext) / (Number of bits in ciphertext). A good cipher should always satisfy an avalanche > 50%.

- By applying a sequences of plain texts with constant key (32 Fs). The result will be as follows:

ID	The plain texts (32 Hex Digits)	Avalanche effect in the original AES	Avalanche effect in the D-AES Phase 1	Avalanche effect in the D-AES Phase 2
1	a95c145dff3ff291307e8582adf0f52f	0.51	0.53	0.57
2	9e90be643882e228c52ce177de8c7a41	0.53	0.5	0.55

3	8dc614d4de94282fdce6d494f3d83a60	0.51	0.55	0.54
4	76eb0cfa51a91dd0603fec8257f0151a	0.52	0.52	0.56
5	5bc37ae4668e0f4ec0648ca87f4ebb05	0.51	0.5	0.53
6	35200f4237a2890b42069ea9b80802d3	0.5	0.56	0.57
7	637ba4a6d25ea088a18d1122bb793e2f	0.51	0.52	0.58
8	6214a2e936da8e2117e25484f8991f4e	0.47	0.55	0.55
9	2437420fad20980906a930519b559834	0.53	0.56	0.55
10	05d2ec4e13149bfa6920fab4419751d4	0.48	0.56	0.54
AVG		**0.5**	**0.53**	**0.55**

Table 5-3: a sequences of plain texts with constant key- Avalanche effect

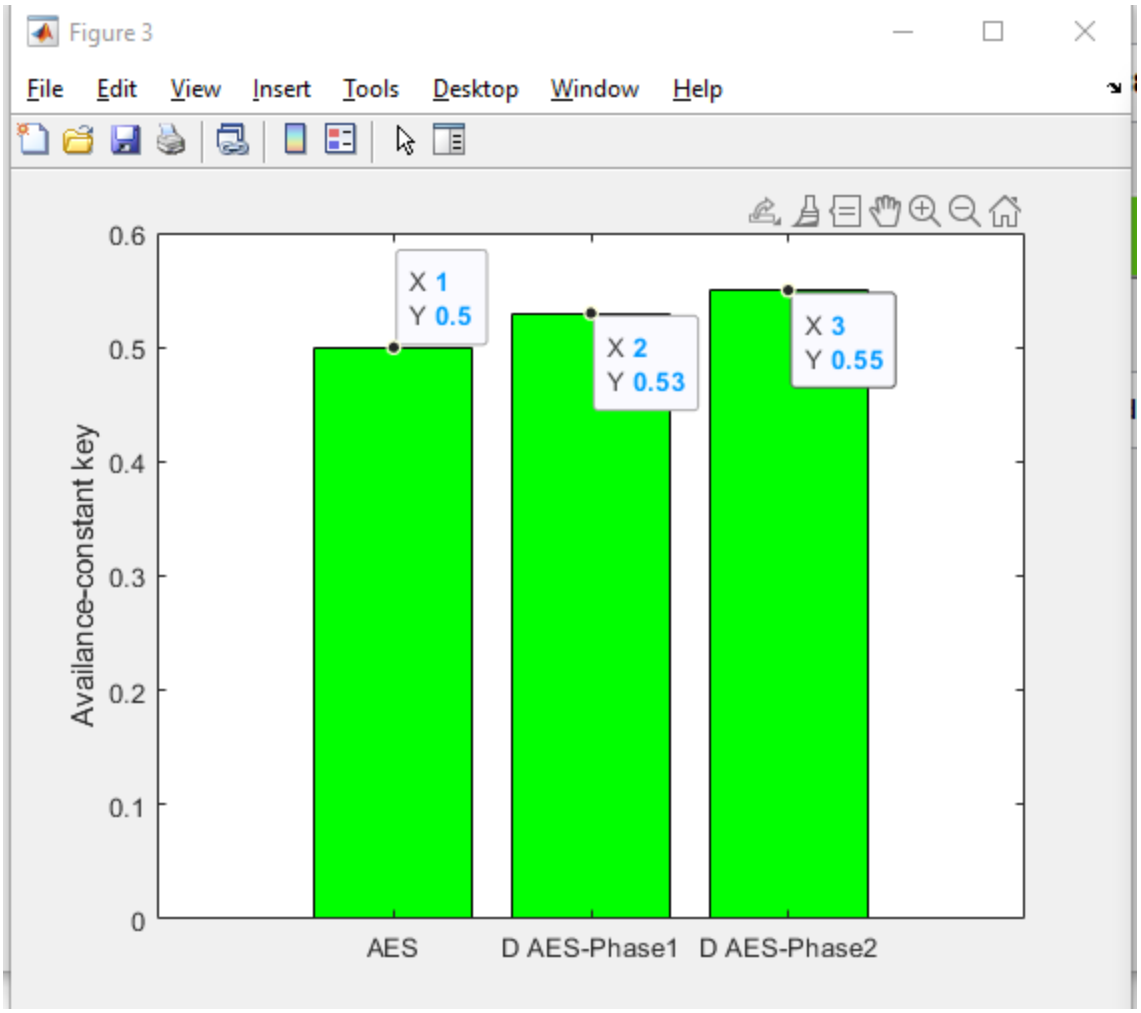

Figure 5-5 : Average Avalanche effect parameter-constant key

-Enhanced Percentage: 5.8% in DAES-phase1- ***constant key.***

- Enhanced Percentage: 9.5% in DAES-phase2- ***constant key.***

-By applying a sequences of keys with constant plain text (32 Fs). The result will be as follows:

ID	The key (32 Hex Digits)	Avalanche effect in the original AES	Avalanche effect in the D-AES Phase 1	Avalanche effect in the D-AES Phase 2
1	2646e60a2ba7dcd2508a6eac28b73010	0.5	0.52	0.57
2	4f04d346dfe774d7e4d80b353eafffc6	0.49	0.57	0.56
3	ff908bc64d4378c8a7b05a8dce7e3dcc	0.52	0.56	0.55
4	f7617a7c96e974dea278f8b40e896ae9	0.51	0.59	0.57
5	8c410243919e54c308b320da151725e5	0.48	0.5	0.59
6	4f361575d0cd17f70b58b53be8b7abf4	0.47	0.57	0.56
7	7c0ce8f90da9fdc1880f04ad31307089	0.49	0.55	0.59
8	8366a53e9a615700c6736b432650154f	0.52	0.56	0.58
9	9daad39ce0abdd349a8fba6994a322cb	0.53	0.54	0.59
10	aa7f2bdb7ab42fc44fb9d69fac7286ec	0.55	0.54	0.54
AVG		**0.5**	**0.55**	**0.57**

Table 5-4: a sequences of plain texts with constant plain text - Avalanche effect

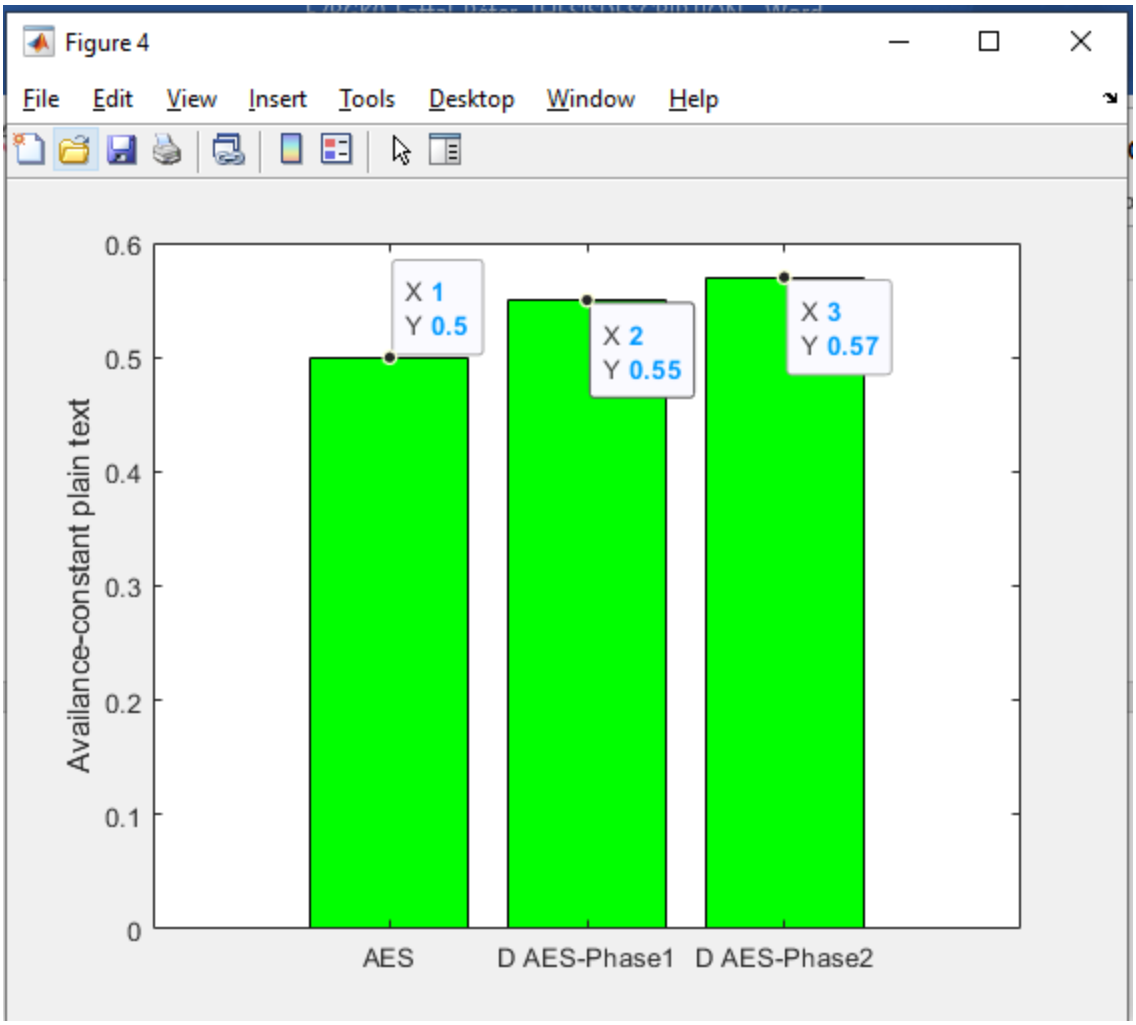

Figure 5-6 : Average Avalanche effect parameter-constant plain text

-Enhanced Percentage: 9.5 % in DAES-phase1-*constant plain text.*

- Enhanced Percentage: 13 % in DAES-phase2- *constant plain text.*

5.3 Balance parameter

Balance parameter depends on number of ones and zeros in cipher text. The best percentage is 50% which means 64 ones and 64 zeros in the cipher text, and it is the most non-linear percentage. [19]

- By applying a sequences of plain texts with constant key (32 Fs). The result will be as follows:

ID	The plain texts (32 Hex Digits)	N.1's t in the original AES	N.1's in the D-AES Phase 1	N.1's in the D-AES Phase 2
1	a95c145dff3ff291307e8582adf0f52f	64	67	62
2	9e90be643882e228c52ce177de8c7a41	77	66	64
3	8dc614d4de94282fdce6d494f3d83a60	68	69	60
4	76eb0cfa51a91dd0603fec8257f0151a	64	67	66
5	5bc37ae4668e0f4ec0648ca87f4ebb05	68	65	72
6	35200f4237a2890b42069ea9b80802d3	79	70	59
7	637ba4a6d25ea088a18d1122bb793e2f	68	60	61
8	6214a2e936da8e2117e25484f8991f4e	65	72	67
9	2437420fad20980906a930519b559834	66	65	71
10	05d2ec4e13149bfa6920fab4419751d4	79	71	67
AVG		**69.8**	**67.2**	**64.9**

Table 5-5: a sequences of plain texts with constant key- Balance parameter

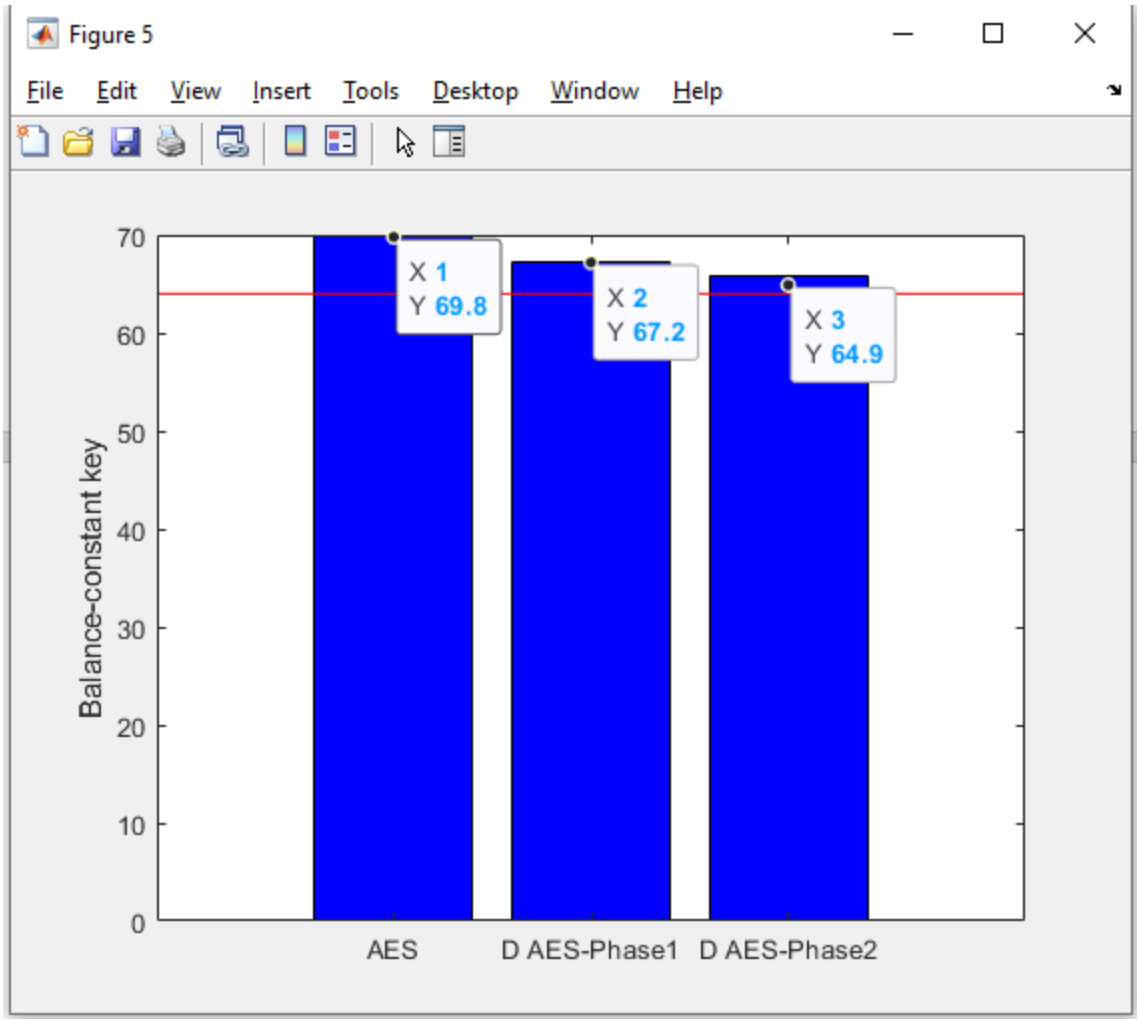

Figure 5-7 : Average Balance parameter-constant key

-Enhanced Percentage: 3.7% in DAES-phase1- ***constant key.***

- Enhanced Percentage: 7.2% in DAES-phase2- ***constant key.***

-By applying a sequences of keys with constant plain text (32 Fs). The result will be as follows:

ID	The key (32 Hex Digits)	N.1's t in the original AES	N.1's in the D-AES Phase 1	N.1's in the D-AES Phase 2
1	2646e60a2ba7dcd2508a6eac28b73010	62	69	72
2	4f04d346dfe774d7e4d80b353eafffc6	65	70	69
3	ff908bc64d4378c8a7b05a8dce7e3dcc	74	77	65
4	f7617a7c96e974dea278f8b40e896ae9	75	65	71
5	8c410243919e54c308b320da151725e5	77	62	61
6	4f361575d0cd17f70b58b53be8b7abf4	75	64	77
7	7c0ce8f90da9fdc1880f04ad31307089	67	75	71
8	8366a53e9a615700c6736b432650154f	64	77	71
9	9daad39ce0abdd349a8fba6994a322cb	79	61	77
10	aa7f2bdb7ab42fc44fb9d69fac7286ec	62	72	59
AVG		**70**	**69.2**	**69.3**

Table 5-6: a sequences of plain texts with constant plain text - Balance parameter

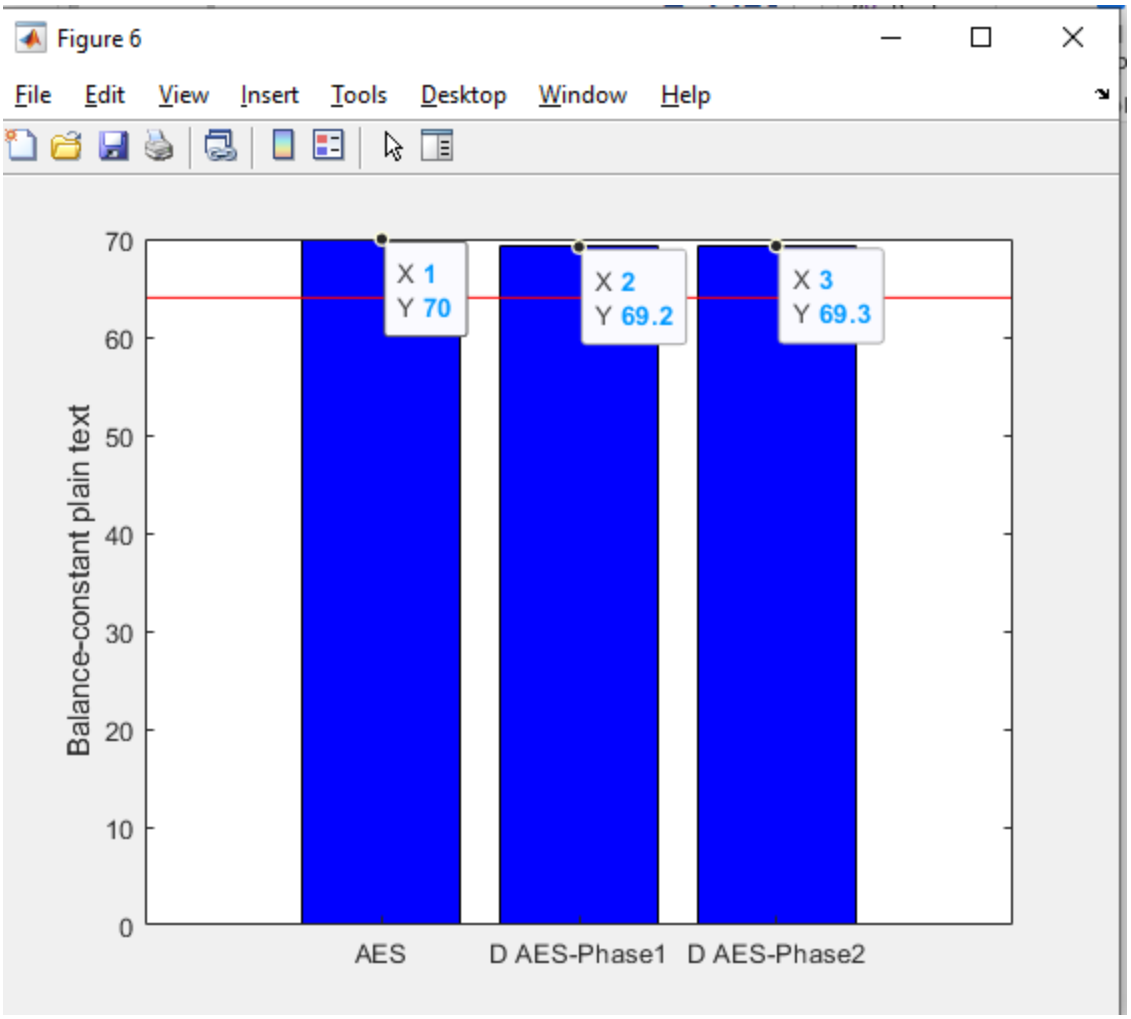

Figure 5-8 : Average Balance parameter-constant plain text

-Enhanced Percentage: 1.1% in DAES-phase1- ***constant plain text***

- Enhanced Percentage: 1% in DAES-phase2- ***constant plain text***

5.4 Conclusion and future work

The enhanced DAES algorithm was implemented using MATLAB program through two phases, the first phase was implemented by modified the key expansion step to be more secure and complicated, the next phase was implemented by adding a modification for shift-rows step. The examined parameters was applied to each of the original AES and DAES-phase1 and DAES-Phase2 by applying a sequence of inputs. The results reveal that both DAES-Phase1 and DAES-Phase2 improved the values of examined parameters over the original AES with greater

enhancement in the final phase DAES-Phase2 than the initial phase DAES-Phase1. Also the results reveal that constant plain text cases has grater enhancement than constant key cases. Overall, the enhanced algorithm DAES aims to provide a highly secure and robust encryption. Evaluating and improving the computational capabilities of the proposed algorithm could be one of the cases used for further research.

List of Tables

List of Figures

6 References

[1] McGinthy, J.M., 2019. Solutions for internet of things security challenges: trust and authentication (Doctoral dissertation, Virginia Tech).

[2] King, J., 2015. A Distributed Security Scheme to Secure Data Communication between Class-0 IoT Devices and the Internet.

[3] Real-Life Use Cases for Edge Computing. https://innovationatwork.ieee.org/real-life-edge-computing-use-cases/, last visited: 05/02/2022

[4] Simmons, G.J., 1979. Symmetric and asymmetric encryption. *ACM Computing Surveys (CSUR)*, *11*(4), pp.305-330.

[5] Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data.

https://www.researchgate.net/publication/317615794_Advanced_Encryption_Standard_AES_Algorithm_to_Encrypt_and_Decrypt_Data, last visited: 03/04/2022

[6] AES encryption and decryption standards.

https://www.researchgate.net/publication/333575801_AES_encryption_and_decryption_standards, last visited: 29/03/2022

[7] Abdullah, A.M., 2017. Advanced encryption standard (AES) algorithm to encrypt and decrypt data. Cryptography and Network Security, 16, pp.1-11.

[8] ADVANCED ENCRYPTION STANDARD. https://www2.rivier.edu/journal/ROAJ-Fall-2010/J455-Selent-AES.pdf, last visited: 02/01/2022

[9] Riyaldhi, R. and Kurniawan, A., 2017. Improvement of advanced encryption standard algorithm with shift row and S. box modification mapping in mix column. *Procedia computer science*, *116*, pp.401-407.

[10] Eddine, G.S., Ouarda, A.S.S.A.S. and Brahim, B.O.U.D.E.R.A.H., High Lightweight Encryption Standard (HLES) as an Improvement of 512-Bit AES for Secure Multimedia.

[11] Kassab, M., Nithya, V. and Mokyed, M.S., 2021, July. HNS Advanced Encryption Standard: An Enhanced Security Approach for IoT Communication. In *Journal of Physics: Conference Series* (Vol. 1964, No. 6, p. 062015). IOP Publishing.

[12]Modified Advanced Encryption Standard Algorithm for Reliable Real-Time Communications. https://iugspace.iugaza.edu.ps/bitstream/handle/20.500.12358/18817/file_1.pdf?sequence=1&isAllowed=y, last visited: 05/11/2021

[13] Singh, A., 2015. A new approach to enhance avalanche effect in aes to improve computer security. *Journal of Information Technology & Software Engineering*, *5*(1), p.1.

[14] Reddy, D.L.K.D.A.R. and Jilani, S.A.K., 2016. Implementation of 128-bit AES algorithm in MATLAB. *Int. J. Eng. Trends Technol.(IJETT)*, *33*(3), pp.126-129.

[15] Advanced Encryption Standard (AES)-128,192, 256.

https://www.mathworks.com/matlabcentral/fileexchange/73412-advanced-encryption-standard-aes-128-192-256, last visited: 17/05/2021

[16] Getting Started with Graphical User Interface in Matlab. https://www.section.io/engineering-education/matlab-graphical-user-interface/, last visited: 28/03/2022

[17] Total Hamming Distance.https://medium.com/geekculture/total-hamming-distance-problem-1b74decd71c9, last visited: 02/05/2022

[18] Avalanche effect. https://en.wikipedia.org/wiki/Avalanche_effect, last visited: 18/02/2022

[19] Imdad, M., Ramli, S.N. and Mahdin, H., 2022. An Enhanced Key Schedule Algorithm of PRESENT-128 Block Cipher for Random and Non-Random Secret Keys. Symmetry, 14(3), p.604.